

المعالجات الدقيقة

Microprocessors

Intel 80186 - Intel 8086 - Z80 - Intel 8085
Pentium - Intel 80486 - Intel 80386 - Intel 80286

البرمجة والمعمارية والمتطلبات

تأليف

أ.د. محمد إبراهيم العدنى

أستاذ بقسم الاتصالات والإلكترونيات

كلية التربية - جامعة حلوان

دار الناشر والتقاويم

المعالجات الدقيقة

Microporocessors

Intel 80186 – Intel 8086 – Intel 8085

Pentium 8 – Intel 80486 – Intel 80386

Intel 80286

البرمجة... وواجهة... والتطبيق

أ.د. محمد إبراهيم العدوسي

أستاذ ورئيس قسم الاتصالات والإلكترونيات

كلية الهندسة – جامعة حلوان

الدار الدولية للاستعلامات الثقافية

مصر

الطبعة الأولى

٢٠٠٠

المعاجن الدقيقة

تأليف

د. محمد إبراهيم العدوسي

رقم الإيداع

2000/3149

I.S.B.N

977-282-076-5

لا يجوز نشر أي جزء من هذا الكتاب أو احتزان مادته بطريقة الاسترجاع أو نقله على أي نحو أو بأي طريقة سواء أكانت إلكترونية أو ميكانيكية أو خلاف ذلك إلا بموافقة الناشر على هذا كتابة ومقدماً.

**حقوق الطبع والاقتباس
والترجمة والنشر محفوظة
للدار الدولية للاستثمارات الثقافية ش.م.م.**

8 إبراهيم العواين - النزهة الجديدة - مصر الجديدة - القاهرة - ج.م.ع.
ص.ب، 5599 هليوبوليس غرب / القاهرة - تليفون: 2957655/2972344 فاكس: (00202) 2957655

المحتويات

الإهداء عرض الكتاب

الفصل الأول : عصر الميكروبروسيسور	1
1-1 عصر الميكروبروسيسور	3
2-1 أين يقع المعالج في داخل الميكروكمبيوتر ؟	5
1-2-1 الذاكرة	6
2-2-1 وحدات الأدخال والأخراج	7
3-2-1 وحدة المعالجة المركزية	8
3-1 ماذا تعنى هذه الألفاظ ؟	8
1-3-1 الميكروكمبيوتر والميكروبروسيسور	8
2-3-1 البرمجة والبناء	9
3-3-1 الأمر وال برنامح	10
4-3-1 لغات البرمجة	10
5-3-1 البت والبait	12
4-1 تمارين	12
الفصل الثاني : البناء المعماري للمعالج	15
1-2 مقدمة	17
2-2 المهام الأساسية المطلوبة من المعالج	17
3-2 أجزاء المعالج الأساسية	18
4-2 المسجلات والعدادات في شريحة المعالج	18
1-4-2 مسجل التراكم accumulator	19
2-4-2 عداد البرنامج program counter	20
3-4-2 مسجل وفاكك شفرة الأوامر instruction register & decoder	20
4-4-2 مسجل الحالة status register	21
1-4-4-2 علم الصفر zero flag	21
2-4-4-2 علم الأشارة sign flag	21
3-4-4-2 علم الحمل carry flag	21
4-4-4-2 علم الباريتى parity flag	22
5-4-4-2 علم الحمل النصفي أو البني	22

22	half carry flag 5-4-2 مسجل مؤشر المكادسة
23	stack pointer register 6-4-2 المسجلات عامة الأغراض
23	5-2 نظرية خارجية على شرائح المعالج
25	address bus 1-5-2 مسار العنوانين
26	data bus 2-5-2 مسار البيانات
26	control lines 3-5-2 خطوط التحكم
27	2-2 شرائح المعالجات ذات 8 بت
30	Intel8085 1-6-2 الشريحة
30	Z80 2-6-2 المعالج
33	2-7 تمارين
 الفصل الثالث : برمجة المعالج	
35	1-3 مقدمة
37	2-3 لغات الحاسب
37	3-3 ما هو الأمر ؟
38	3-4 ما هو البرنامج ؟
39	3-5 كيف يقوم المعالج بتنفيذ البرنامج ؟
40	3-6 طريقة كتابة البرنامج للمعالج
40	1-6-3 الشفرات الثنائية
40	2-6-3 الشفرات المستعرية
42	3-6-3 الشفرات الحرفية mnemonics codes
44	3-7 اللغات ذات المستوى العالى high level languages
45	3-8 خطوات كتابة برنامج بلغة الأسمبلی
48	3-9 تمارين
 الفصل الرابع : برمجة المعالج Intel8085	
51	1-4 مقدمة
53	2-4 مجموعة أوامر الانتقال Transfer instructions
53	1-2-4 الأمر MOV
55	2-2-4 الأمر MVI
57	3-2-4 الأمر LXI
60	4-2-4 الأمران STA و LDA
61	5-2-4 الأمران LHLD و SHLD

62	3-4 تمارين
63	4-4 مجموعة أوامر الحساب arithmatic instructions
64	1-4-4 الأمران ADD و SUB
66	2-4-4 الأمران ADI و SUI
67	3-4-4 الأمران ADC و SBB
69	4-4-4 الأمران INR و DCR
71	5-4-4 الأمران INX و DCX
71	5-4 تمارين
73	6-4 مجموعة أوامر القفز jump instructions
73	1-6-4 القفز غير المشروط unconditional jump
73	2-6-4 القفز المشروط conditional jump
75	7-4 مهمة أخرى للأسمبلر
75	1-7-4 الأمر والمعاملات
77	2-7-4 التعليق comment
78	3-7-4 العلامة label
79	8-4 أوامر الأدخال والأخراج input output instructions
83	9-4 مجموعة أوامر المنطق logic instructions
85	10-4 كيفية الاتصال بالذاكرة Memory addressing
85	1-10-4 الطريقة المباشرة direct method
85	2-10-4 الطريقة غير المباشرة indirect method
95	11-4 تمارين

99	الفصل الخامس : برمجة المعالج Z80
101	1-5 مقدمة
101	2-5 مجموعة أوامر الأنقال transfer instructions
101	1-2-5 نقل معلومة من مسجل الى مسجل آخر
102	2-2-5 تحميل مسجل بثابت او معلومة فورية
105	3-2-5 نقل معلومة من مسجل الى الذاكرة والعكس
106	3-2-5 الطريقة المباشرة direct addressing
	2-3-2-5 الطريقة غير المباشرة indirect addressing
107	3-3-2-5 طريقة الفهرسة indexed addressing
109	3-5 تمارين
110	4-5 مجموعة أوامر الحساب arithmatic instructions

110	1-4-5 الأمران ADD و SUB
113	2-4-5 الأمران ADC و SBC
115	3-4-5 الأمران INC و DEC
115	4-4-5 العمليات الحسابية على أزواج المسجلات
117	5-4-5 أمر المقارنة compare instruction
118	5 تمارين
119	6-5 مجموعة أوامر القفز jump instructions
119	1-6-5 القفز غير المشروط unconditional jump
120	2-6-5 القفز المشروط conditional jump
120	3-6-5 القفز النسبي relative jump
121	7 مهمة أخرى للأسمبلر
121	8-5 أوامر الأدخال والأخراج input output instructions
122	1-8-5 أوامر الأدخال input instructions
123	2-8-5 أوامر الأخراج output instructions
123	9-5 مجموعة أوامر المنطق logic instructions
141	10-5 تمارين
145	الفصل السادس : المعالج من البداية حتى النهاية
147	1-6 مقدمة
147	6-2 الجمع الثنائي binary addition
148	1-2-6 دائرة نصف المجمع half adder
149	2-2-6 دائرة المجمع الكامل full adder
150	6-3 الطرح الثنائي binary subtraction
154	6-4 وحدة الحسابي والمنطق arithmatic and logic unit
157	5-6 مسجل التراكم accumulator register
161	6-6 اضافة ذاكرة للمعالج الافتراضي
165	6-7 تمارين
167	الفصل السابع : أساسيات مواجهة المعالج
169	1-7 مقدمة
169	7-2 فصل buffering خطوط المعالج
169	1-2-7 ماذا نعني بكلمة فصل ؟
170	2-2-7 متى نحتاج لفصل خطوط المعالج ؟
172	7-3 البوابات ثلاثة المنطق tristate logic gates

174	1-3 بوابات المنطق الثنائي
175	2-3 بوابات ثلاثة المنطق
177	4- الماسك latch
178	5- بعض الشرائح التي تستخدم في فصل المسارات
178	1- الشريحة 74244 عازل ثمانى ثالثى المنطق
	2- الشريحة 74245 فاصل ذو ثمان بิตات ثالث
179	الأتجاه ثلاثة المنطق
181	3- الشريحة 74374 فاصل ماسك ذو ثمان بيتات
182	7- تمارين
 الفصل الثامن : فصل مسارات المعالج	
183	1- مقدمة
185	2- لماذا مسار العنوان ؟
186	3- لماذا مسار التحكم ؟
188	4- تهيئة مسارات المعالج 8085 لعملية المواجهة
190	1-4 مسار العنوانين للمعالج 8085
191	2-4 مسار البيانات للمعالج 8085
192	3-4 مسار التحكم للمعالج 8085
194	5- تهيئة مسارات المعالج Z80 لعملية المواجهة
196	6- تمارين
 الفصل التاسع : مواجهة الذاكرة memory interfacing	
203	1- مقدمة
205	2- أساسيات بناء ذاكرة الحاسب
206	3- كيف سنوصل الذاكرة على المعالج ؟
210	1-3- مثال توضيحي
210	2-3- نظام بلوكتس الذاكرة
213	3-3- بناء البلوكتس من شرائح
220	4- تمارين
 الفصل العاشر : الأدخال وال而出	
221	1- مقدمة
223	2- طرق ارسال واستقبال المعلومات الرقمية
225	3- الطريقة الأولى من طرق الأدخال وال而出

227	باستخدام الأمرين IN و OUT
228	1-3-10 دائرة بوابة الأخراج output port
234	2-3-10 دائرة بوابة الأدخال input port
	4-10 الطريقة الثانية من طرق الإدخال والإخراج
236	باستخدام خريطة الذاكرة memory map
238	1-4-10 دائرة بوابة الأخراج باستخدام خريطة الذاكرة
239	2-4-10 دائرة بوابة الأدخال باستخدام خريطة الذاكرة
241	5-10 البوابات القابلة للبرمجة
242	1-5-10 تركيب الشريحة 8255A
244	2-5-10 برمجة الشريحة 8255A
248	3-5-10 حالات modes تشغيل الشريحة 8255A
249	1-3-5-10 الحالة صفر mode zero
252	2-3-5-10 الحالة واحد mode one
258	3-3-5-10 الحالة اثنين mode two
258	6-10 تمارين
 الفصل الحادى عشر : التحكم فى إشارة مرور	
261	1-11 مقدمة
263	2-11 تركيب الدائرة
263	1-2-11 مثال توضيحي
264	3-11 الأطراف الأخرى للمعالج 8085
266	1-3-11 أشارات الترامن clock
266	2-3-11 بدأ و إعادة التشغيل
267	3-3-11 الطرفان HOLD و HLDA
269	4-3-11 الطرف READY
270	5-3-11 طرفى الحالة S0 و S1
271	6-3-11 أطراف المقاطعة
272	7-3-11 الطرفان SID و SOD
272	4-11 الأطراف الأخرى للمعالج Z80
272	1-4-11 أطراف المقاطعة
273	2-4-11 الطرف RFSH
273	3-4-11 الطرف HALT
273	4-4-11 الطرف WAIT
274	5-4-11 الطرف M1

274	6-4-11 CLK
274	5-11 إشارات المرور
282	6-11 تمارين
283	الفصل الثاني عشر : البرامج الفرعية subroutines
285	1-12 مقدمة
285	2-12 ما هو البرنامج الفرعى ؟
287	3-12 كيف يعود المعالج الى نفس المكان الذى خرج منه ؟
291	4-12 حساب أزمنة التأخير
294	5-12 تمارين
297	الفصل الثالث عشر : المقاطعة interrupt
299	1-13 مقدمة
299	2-13 طريقة طرق الأبواب لخدمة الأجهزة المحيطة polling
301	3-13 المقاطعة
302	4-13 مقاطعة المعالج 8085
303	1-4-13 الخطوط RST5.5 و RST6.5 و RST7.5
312	2-4-13 TRAP الخط
313	3-4-13 INTR الخط
314	4-4-13 كيف يتم تحديد العنوان الذى سيتم القفز اليه فى حالة المقاطعة INTR ؟
315	5-13 مقاطعة المعالج Z80
315	1-5-13 الخط NMI
317	2-5-13 الخط INT
321	6-13 تمارين
323	الفصل الرابع عشر : التركيب الهيكلى للمعالج intel/8086/8088
325	1-14 مقدمة
326	2-14 نظرة داخلية على محتويات المعالج 8086/8088
327	3-14 نظرة تفصيلية على مسجلات المعالج 8086/8088
329	1-3-14 المسجلات عامة الأغراض
332	2-3-14 المسجلات الخاصة
332	4-14 تجزئء الذاكرة memory segmentation
335	1-4-14 مسجل تجزئء البرامح CS

335	2-4-14 مسجل تجزيء البيانات DS
335	3-4-14 مسجل تجزيء المكملة SS
336	4-4-14 مسجل التجزيء الإضافي ES
336	5-4-14 مسجل الأعلام SR
338	5- طرق العنونة addressing modes
339	1-5-14 عنونة المسجل
339	2-5-14 العنونة الفورية
340	3-5-14 العنونة المباشرة
340	4-5-14 العنونة غير المباشرة
341	5-5-14 عنونة القاعدة زائد الفهرسة
341	6-5-14 العنونة النسبية
342	6-14 تمارين
345	الفصل الخامس عشر : برمجة المعالج intel8086/8088
347	1-15 مقدمة
347	2- خطوات كتابة وتنفيذ برامج لغة التجميع
348	3-15 مكونات برنامج الأسمبل
353	4- اوامر لغة الأسمبل
353	5-15 مجموعة اوامر الانتقال
356	6-15 الدببور debugger
356	6-1 اظهار محتويات المسجلات
357	6-2 عرض اوامر الأسمبل ابتداء من عنوان معين
358	6-3 عرض محتويات جزء من الذاكرة
358	6-4 تنفيذ البرنامج حتى عنوان معين
359	6-5 متابعة تنفيذ البرنامج عن طريق تنفيذ عدد n من الخطوات
359	6-6 تغيير محتويات عنوان في الذاكرة
360	6-7 الخروج من الدببور
360	7-15 تمارين
361	8-15 اوامر القفز
361	8-1 القفز غير المشروط
361	8-2 القفز المشروط
363	8-3 الأمر Loop
364	9-15 أول خطوات التعامل مع الذاكرة

364	1-9-15 الطريقة المباشرة
364	2-9-15 الطريقة غير المباشرة
366	أوامر الحساب 10-15
374	أوامر المنطق 11-15
376	أوامر الإزاحة والدوران 12-15
383	تمارين 13-15

385	الفصل السادس عشر : مواجهة المعالج 8086/8088
387	1-16 مقدمة
387	2-16 الوظائف المختلفة لأطراف الشريحة 8086/8088
393	1-2-16 نبضات الساعة
393	3-16 عزل مسارات المعالج 8086
394	4-16 مواجهة الشريحة 8086/8088 مع الذاكرة
399	5-16 الإدخال والإخراج من وإلى المعالج 8086/8088
402	6-16 شريحة مواجهة لوحة المفاتيح القابلة للبرمجة 8279
404	7-16 المؤقت القابل للبرمجة 8254
411	8-16 الاتصالات القابلة للبرمجة 8251
413	9-16 الاتصال المباشر مع الذاكرة 8237A
414	10-16 المواجهة مع المعالجات الحاسوبية المساعدة 80x87
415	11-16 تمارين

417	الفصل السابع عشر : ثم ماذا ؟ What else ?
419	1-17 مقدمة
419	2-17 المعالج 80186
421	1-2-17 أطراف المعالج 80186
425	2-2-17 برمجة المعالج 80186
425	3-17 المعالج 80286
426	1-3-17 التركيب الهيكلي للمعالج 80286
428	4-17 المعالج 80386
428	1-4-17 التركيب الهيكلي للمعالج 80386
428	2-4-17 تنظيم الذاكرة للمعالج 80386
431	3-4-17 نظام الإدخال والإخراج في المعالج 80386
431	4-4-17 أطراف المعالج 80386
433	5-4-17 مسجلات المعالج 80386

434	17-5 الذاكرة المخبأة Cache memory
435	17-6 المعالج 80486
436	17-7 انسبيافية الأوامر Instruction pipelining
441	17-8 سلسلة معالجات بنتيم Pentium Processors
443	17-9 المعالج بنتيم برو Pentium Pro Processor
445	17-10 تمارين
447	الملحق الأول : الحساب الرقمي
455	مراجع الكتاب

إهداء
إلى اللغة العربية

ظموك فقالوا إنك لست لغة علوم . . . مع أنك والله مثل الإنجليزية والكثير من اللغات الأوربية (أربعة وعشرون حرفاً أو تزيد قليلاً) . . . ويتكلّم بك عدد لا يأس به مثلها تماماً . . . أين أنت من اليابانية التي لها أربعة آلاف حرفاً أو تزيد . . . ولكنها قوية بقوة أهلها . . . وعزيزه بعزتها عند أهلها . . . ولكن يكفيك فخراً أنك لغة القرآن الكريم المحفوظ من قبل الله عز وجل . . . وبذلك ضمنت الحفظ والصون إلى الأبد . . . كان من الممكن أن تكوني في عالم النسيان في ظل عقد الخواجہ التي يعيشها أبنائك الآن لو لا حفظك بالقرآن الكريم . . .

حَمْدًا لِلَّهِ

عرض الكتاب

عندما ظهر الترانزistor في أواخر الأربعينات الميلادية أحدث ما يشبه الثورة في مجال الإلكترونيات بحيث أنه أصبح الوحدة الأساسية لبناء أي دائرة إلكترونية. بظهور مكبر العمليات operational amplifier في منتصف السبعينات دائرة تكاملية رخيصة التكاليف ، صغيرة الحجم ، سلبت الأضواء من الترانزistor ودخل مكبر العمليات كوحدة أساسية جديدة في الكثير من التطبيقات والدوائر الإلكترونية . في منتصف السبعينات تألف نجم الميكروبروسيسور (المعالج) وانتشر استخدامه في الكثير من الدوائر والأنظمة الإلكترونية وأنظمة التحكم ، حتى أنك تجده الآن في السيارة يتحكم في الكثير من متغيراتها ، وفي الصاروخ يتحكم في توجيهه ، وفي الميكروكمبيوتر يتحكم في تشغيله ، بل في لعبة الطفل يتحكم في الكثير من أدائها .

موضوع المعالجات من الموضوعات التي يصعب جداً أن تجمع أو تنتهي في مؤلف أو كتاب واحد يرضي جميع القراء وذلك نظراً للتشعبات الكثيرة التي يمكن أن يتشعب إليها هذا الموضوع ، وما من قارئ يقرأ كتاباً في هذا الموضوع إلا ويقول ليت المؤلف أضاف كذا وحذف كذا تبعاً لنظرته واهتماماته الخاصة . هناك مثلاً من القراء من يهتم بالبرمجة فقط software ولا يهتم كثيراً بموضوعات المواجهة والبناء hardware ، كما أن هناك العكس ، فمن القراء من يهتم كثيراً بالمواجهة والبناء على حساب البرمجة .

لقد رأينا في هذا الكتاب أن يفي بقدر الإمكان باحتياجات الكثير من القراء ، فهذا الكتاب يقدم للقارئ الكثير من شرائح المعالجات الشائعة الاستخدام ذات 8 و 16 و 32 بت وذلك حتى يجد أي مستخدم لهذه الشرائح ما يفيده وتكون الفرصة متاحة لمن يرغب في المقارنة بين أكثر من شريحة ويشهد تطور المعالجات عبر الأجيال المختلفة . يحتوى الكتاب أيضاً على فصل خاص ببرمجة كل شريحة على حدة من الشرائح 8085 و Z80 و 8086 من خلال شرح ميسر وأمثلة عديدة على أوامر كل شريحة وذلك في الفصول 4 و 5 و 15 وذلك بعد أن سبقت هذه الفصول بفصل كمقدمة عن المعالجات بصفة عامة وما هو دورها في الميكروكمبيوتر وهو الفصل الأول ، ثم تلا ذلك فصل خاص بالتركيب الداخلى للميكروبروسيسور بصفة عامة على ضوء الوظائف المطلوبة منه وهو الفصل الثاني حيث انتهى هذا الفصل بعرض التركيب الداخلى للمعالجين 8085 و Z80 ، وبعد ذلك قدم الفصل الثالث عرضاً للغات البرمجة وكيفية برمجة المعالج . كتطبيق على الفصول الخمسة الأولى من الكتاب يحتوى الفصل السادس على عملية بناء لمعالج افتراضى hypothetical من البداية (ابتداء من دائرة نصف

المجمع) ثم الارتفاع بهذه الدائرة إلى أن يتم الحصول على وحدة حساب ومنطق ثم توصيل هذه الوحدة مع مركم accumulator ثم توصيلها مع الذاكرة وعمل قائمة أوامر خاصة بهذه الوحدة لبرمجتها . بذلك ينتهي تقريراً جزءاً منهم من الكتاب وهو الجزء الخاص بالمقدمة وتركيب المعالجات ذات 8 بت وبرمجتها .

يبدأ بعد ذلك الفصل السابع الذي يحتوى على بعض الأساسيات التي يجب الإلمام بها قبل الدخول في عمليات المواجهة مع المعالج ، مثل عمليات الفصل أو العزل buffering ومتي تحتاج إليها والمنطق الثالثي أيضاً ولماذا تحتاجه مع عرض بعض الشرائح التي تستخدم في ذلك . لمواجهة المعالج مع شرائح الذاكرة مثلاً لابد من تهيئة المسارات الثلاثة (العناوين والبيانات والتحكم) والحصول عليها في صورة مناسبة لأى عملية مواجهة ، حيث يحتوى الفصل الثامن على ذلك بالتفصيل . بعد عملية التهيئة للمسارات الثلاثة في الفصل الثامن يقدم الفصل التاسع كيفية توصيل الذاكرة على المعالج ثم يقدم الفصل العاشر كيفية توصيل المعالج على بوابات أو منافذ الإدخال والإخراج وذلك بالطرق المختلفة . بانتهاء الفصل الحادى عشر يتم الانتهاء من دراسة الأجزاء الرئيسية اللازمة لبناء دوائر التحكم التي تستخدم المعالج ولذلك فإن الفصل الحادى عشر يقدم مثلاً متكاملاً بجزئي البرمجة والبناء لنظام التحكم فى إشارة مرور فى تقاطع رباعى وذلك كمثال يمكن تطبيقه فى أي تطبيق آخر . البرمجة المتقدمة للمعالج تحتاج لبعض الموضوعات التي رأينا أن يفرد لكل منها فصل خاص بها ، من هذه الموضوعات موضوع البرامج الفرعية subroutines والذي أفرد له الفصل الثانى عشر ، ثم موضوع المقاطعة interrupt وقد خصص له الفصل الثالث عشر .

المعالجات ذات 16 بت ويمثلها المعالج intel8086 قد فرضت نفسها على السوق فترة ليست بالقليلة ممثلة في الحاسوبات XT . لذلك قد أفردنا لها أكثر من فصل ، فالفصل الرابع عشر يقدم تفاصيل التركيب الهيكلى لهذا المعالج ، والفصل الخامس عشر يقدم تفاصيل برمجة هذا المعالج حيث أن لغة التجميع لهذا المعالج تعتبر الأساس لكل المعالجات التالية . الفصل السادس عشر يقدم كيفية مواجهة هذا المعالج مع الدوائر الخارجية مثل الذاكرة وبوابات الإدخال والإخراج . وأخيراً يقدم الفصل السابع عشر فكرة مختصرة ولكننا نعتقد أنها كافية عن المعالجات 80186 و 80286 و 80386 و 80486 وكذلك عائلة معالجات بنظام الشهير في السوق هذه الأيام والتي تتوالى إصداراتها حيث نفاجأ بإصدار جديد منها كل ستة شهور تقريراً .

هذا الكتاب ككتاب دراسي text book يمكن تدريسه للمبتدئين في تعلم موضوع المعالجات على فصلين دراسيين متsequين (ساعتين لكل فصل أو 4 ساعات في فصل واحد) حيث يدرس في الفصل الدراسي الأول المقدمة والفصوص الخاصة ببرمجة المعالج والمعالج الافتراضي وفصل البرامج الفرعية وكذلك مبادئ

مواجهة المعالجات والتطبيق على ذلك بدراسة فصل المواجهة مع الذاكرة وبوابات الإدخال والإخراج . في الفصل الدراسي الثاني يتم تدريس الفصل الخاص المقاطعة واستخدام المعالج في عمليات التحكم المختلفة أو في بناء نظام ميكروكمبيوتر بسيط وبعدها يتم الانتقال إلى المعالجات ذات 16 بت لدراسة تركيبها الداخلي وبرمجتها ومواجهتها ثم يتم الانتقال إلى المعالجات الأخرى لأخذ فكرة سريعة عنها ومقارنتها بما سبقها من معالجات .

لابد أن يصاحب هذا المقرر ساعتين للمعمل أسبوعيا يقوم الطالب فيها بتطبيق كل ما تمت دراسته من برامج أو دوائر من خلال بعض التجارب التي توضع بدقة بحيث تتبع الدراسة النظرية للموضوع وتغطي جانبي البرمجة ودوائر مواجهة المعالج .

لكى تتم الفائدة يجب أن يكون القارئ لهذا الكتاب قد درس قدرا كافيا من الإلكترونيات الرقمية ابتداء من البوابات والدوائر المنطقية ، تبسيطها وطرق بناؤها ، ودوائر الجمع adders ، والمشفرات encoders ، والمنتخبات multiplexer ثم القلابات بأنواعها flip flops ومسجلات الإزاحة shift registers والعدادات counters بأنواعها . كذلك فإنه من الأفضل (وليس بضرورة) أن يكون القارئ قد درس مقدمة الحاسوب وألم فيه بموضوعات الخوارزميات Algorithms وخرائط التدفق أو مخططات السير Flow charts والحلقات Loops والقفز Jump ولا يهم أن يكون ذلك بأى لغة من لغات البرمجة ذات المستوى العالى حيث يتساوى فى ذلك الباسيك أو الباسكال أو C.

يمكن للقارئ المهتم بأى نوع من أنواع المعالجات التى تم تناولها فى هذا الكتاب أن يتتبع الدراسة والتطبيق على هذا النوع فقط دون عناء كبير ودون أن يكون مضطرا لقراءة الكثير عن المعالجات الأخرى التى لا تهمه إن أراد ذلك ، حيث قد تم مراعاة ذلك فى خلال هذا الكتاب بقدر الإمكان مع العلم أن هناك فرصولا تم عرض معظمها بصورة لا تعتمد على المعالج المستخدم مثل فصول مواجهة الذاكرة والإدخال والإخراج .

من الصعوبات التى واجهتنا فى إعداد هذا الكتاب والتى من المتوقع أنها تواجه أي شخص مهتم أو حريص على نشر وترجمة العلوم باللغة العربية هى ترجمة المصطلحات العلمية فى هذا المجال . إننا نقول أنها صعوبات ليس لأن اللغة العربية عاجزة عن إيجاد لفظة عربية تؤدى معنى المصطلح ولكن لأن لفظ المصطلح الأجنبى قد فرض نفسه علينا نحن المشتغلين فى هذا المجال بحيث أصبح من الصعب أن نفلت منه وذلك راجع بالطبع للسبق الذى حققه الناطقون بلغة المصطلح ، ولذلك فقد حاولنا استخدام صورة المصطلح الأجنبية مكتوبة باللغة العربية والإنجليزية مع الترجمة العربية لمعنى المصطلح وذلك حتى نتيح للقارئ فرصة التعرف على المصطلح بلغته الأصلية ، فمثلا كلمة

قد شاعت ترجمتها باللغة العربية بالمعالج الدقيق وأحياناً المعالج الصغرى ونحن استخدمنا الترجمة الأولى وفي معظم الأحيان نستخدم كلمة معالج فقط أو حتى كلمة بروسيسور أو ميكروبروسسور كما يشيع النطق بها حتى تعم الفائدة ولا ننسى المصطلح الأجنبي .

وأخيراً وقبل أن أترك هذا المقام لابد من تقديم كلمات شكر وعرفان لكل من ساعد ولو بالتشجيع في إنجاز هذا الكتاب الذي أخذ الكثير من الجهد . أخص بالشكر هنا الأستاذ الدكتور فريد عبد العزيز طلبة الأستاذ بكلية الهندسة جامعة عين شمس وعميد كلية التعليم الصناعي بالقاهرة الذي تابع وراجع وأبدى الآراء الصائبة بالذات في المراحل الأولى من الكتاب . كذلك أتوجه بالشكر إلى زملاء لى في الكلية التقنية ببريدة بالمملكة العربية السعودية كانوا وراء فكرة التفكير في وضع هذا الكتاب باللغة العربية وأخص بالذكر منهم المهندس صالح الشبعان وأخرون كثيرون . أتوجه بالشكر أيضاً إلى زميلي د.هشام عبد المنعم كشك الذي مارس تدريس هذا الكتاب وكانت له الكثير من الآراء والتوجيهات . كذلك أتوجه بالشكر إلى الكثير من الطلاب بقسم الإلكترونيات بهندسة حلوان الذين مارسوا معهم تدريس هذه المحتويات لمدة خمس سنوات وكم كنت أتلقى منهم التوجيهات والآراء الصائبة . وأخيراً أخص بالشكر والعرفان الذي لمن أنساه لزوجتي وأولادى الذين صبروا على كثيراً وتحملوا مني الكثير على طول فترات انشغالى بإعداد هذا الكتاب .

إن تجربة تأليف كتاب باللغة العربية ليست بالتجربة البسيطة ولكنها تجربة صعبة تحتاج الكثير من الجهد والوقت والتشجيع وإقناع الآخرين بجدوى وفائدة الكتابة باللغة العربية في المجالات العلمية . ومن هنا أوجه نداء إلى كل الزملاء أعضاء هيئات التدريس في الجامعات المصرية والعربية ، إذا كان كل منا يفهم في تخصصه ويجيده فماذا يمنعه من كتابة أفكاره بلغته الأم ؟ والله إن الفائدة لعظيمة من وراء أن يجد الطالب مرجعاً باللغة العربية يتساوى مع أكبر المراجع الأجنبية في الموضوع ، وإن ذلك من واقع تجربى الشخصية في هذا المجال على مدى خمس سنوات على الأقل .

المؤلف

الفصل الأول

عصر الميكروبيوسور

Century of Microprocessor

١-١ عصر الميكروبروسيسور

Century Of Microprocessor

لقد كان للتقدم النشط في علوم الإلكترونيات والسرعة الهائلة التي يمكن أن تتفذ بها العمليات الحسابية والتعقيد الذي وصلت إليه عملية بناء الشرائح الإلكترونية (الدواير التكاملية) الأثر الكبير في جميع نواحي الحياة وعلى الكثير من العلوم المختلفة . كما أن الطفرة الأخيرة التي حدثت في علوم الحاسوبات يرجع الفضل فيها أساسا إلى التقدم في علوم الإلكترونيات والتكنولوجيا الحديثة والمتطرفة في تصنيع الدواير التكاملية Integrated circuits ، فما هي الدائرة التكاملية إذن؟ لكي نعرف ما هي الدائرة التكاملية تعال نرجع إلى الوراء في التاريخ وبالتحديد في بداية الخمسينيات عندما تم اكتشاف الترانزستور . في هذا الوقت كانت الدواير الإلكترونية تبني أو تصمم باستخدام الصمامات المفرغة التي كان منها ما يكفي الترانزستور ومنها ما يكفي الدايمود على سبيل المثال وكان أي صمام من هذه الصمامات عبارة عن أسطوانة زجاجية مفرغة من الهواء يبلغ قطرها حوالي ثلاثة سنتيمترات وارتفاعها حوالي سبعة سنتيمترات وكانت هذه الصمامات تحتاج لتشغيلها إلى فرق جهد مستمر d.c عالي يبلغ فوق 200 فولت ، ولذلك كانت هذه الصمامات تشع الكثير من الحرارة مما كان يتطلب الكثير من وسائل التبريد لها . لذلك كانت جميع الأجهزة الإلكترونية في هذا الوقت تعرف بـ بكر حجمها ، فلما أن تخيل مثلاً أن جهاز حاسب شخصي من أبسط الأجهزة المعروفة الآن ربما كان يشغل حجرتين كاملتين متواسطتي الحجم لو أنه بنى بهذه الصمامات .

باكتشاف أشباه الموصلات وظهور الترانزستور أخذت أحجام الدواير الإلكترونية والفراغ الذي تشغله في الانكماش ، ومنذ ذلك الحين بدأت عجلة التطور في بناء الدواير الإلكترونية في الدوران وأصبح المصممون لا يكتفون ببناء ترانزستور واحد على نفس شريحة شبه الموصل ولكن بدعوا في وضع أكثر من ترانزستور على نفس القطعة ، ثم أضافوا لهذا العدد من الترانزستورات بعض المكونات الأخرى مثل المقاومات والمكثفات ، ثم قاموا بتوصيل هذه المكونات مع الترانزستورات الموجودة على نفس الشريحة للحصول على دائرة إلكترونية تؤدي وظيفة معينة ، هذه الدائرة الإلكترونية المبنية على شريحة واحدة لأداء هدف أو وظيفة معينة هي ما يسمى بالدائرة التكاملية . في بداية السبعينيات كان كل ما تمكنت منه التكنولوجيا في ذلك الوقت هو بناء أو تجميع حوالي عشرة ترانزستورات على نفس الشريحة واستخدمت هذه في بناء دواير البوابات المنطقية مثل بوابة AND وبوابة OR وبوابة NOT وغيرها وسميت هذه الدواير بـ دواير التكامل الصغير (SSI) .

بعد ذلك أخذت تكنولوجيا بناء الدوائر التكاملية في التطور السريع حيث تمكّن المصممون من زيادة كثافة المكونات على نفس الشريحة فظهرت الدوائر ذات التكامل المتوسط (MSI) والتى منها على سبيل المثال دوائر العدادات counters ومسجلات الإزاحة shift registers والكثير من المكبرات التماضية analog amplifiers المتعددة الأغراض ، ولم يقف الأمر عند هذا الحد بل ظهرت بعد ذلك الدوائر عالية التكامل Large Scale Integration (LSI) والتى منها شرائح الذاكرة memory وشرايح المعالجات بجيليها الأول والثانى ، ولم يقف الأمر عند هذا الحد أيضاً بل ظهرت بعد ذلك الدوائر التكاملية الفائقة التكامل (VLSI) والتى منها بعض شرائح الذاكرة والأجيال الأخيرة من شرائح المعالجات والتى منها الجيل الثالث والرابع ، ولذلك أن تخيل الآن أن عدد الترانزستورات على الشريحة الواحدة لا تتعدي مساحتها السنتمتر المربع الواحد قد فاق عدده ملايين من الترانزستورات على نفس الشريحة ، فالمعالج بتقنية برو (آخر أجيال المعالجات هذه الأيام) Pentium Pro يحتوى على 5.5 مليون ترانزستور . وبعلم الله وحده ما سيأتى لنا به المستقبل القريب وإلى أين سيصل هذا العقل البشري ؟ هذه النعمة التي دائمًا يحاول الإنسان تقليديها ولكنه دائمًا سيخفق في تصنيعها!!

لقد كان ظهور المعالج هو السبب في الطفرة الأخيرة التي ظهرت في علوم الحاسوب والذى تركت آثارها بالتأل على جميع العلوم الأخرى بل وأوجدت أو أحيت علوماً كانت على وشك النسيان بسبب المقدرة المحدودة على معالجة وتخزين البيانات في تلك الأوقات ، فمن كان يتخيّل مثلاً أن عملية التحكّم في أنظمة الطاقة الكهربائية ابتدأء من توليدتها وانتهاء بتوزيعها على المشتركين من الممكن أن يلعب الحاسوب دوراً كبيراً فيها ، جميع المصانع الآن لا تخلو من كومبيوتر يتحكم في أعقد العمليات الصناعية فيها ، بل إننا إذا انتقلنا إلى المجال الطبي ودخلنا حجرة العمليات لوجدنا الغالية العظمى من أجهزتها الآن تستخدّم الحاسوب . إن الحاسوب الآن دخل جميع نواحي الحياة فمن كان يتخيّل أن يستخدم المعالج في التحكّم في خلط الهواء بالبنزين بل ومراقبة الكثير من أداء السيارة وأجزائها . إنك من الممكن أن تشتري لعبة لطفالك الآن ليُلعب بها فتفاجأ بأن بداخلها معالج يتحكم فيها . إن علم الذكاء الصناعي Artificial intelligence وعلوم الكلام ، (التعرّف عليه وتوليده) ، وعلوم الروبوتات كل هذه تعتبر قليلاً من كثير من العلوم التي ما كانت ستصل إلى ما وصلت إليه الآن لولا ظهور المعالج . لذلك فإنني أرى أن يسمى هذا العصر فعلاً بعصر (الميكروبروسيسور) المعالجات أسوة بعصر البخار وعصر الكهرباء التي مررت بها البشرية سابقاً .

إن نظرية مبسطة على شريحة المعالج ستجد أنها كباقي الشرائح بـل والأجهزة الإلكترونية ، لكي تتمكن من استخدامها لابد من قراءة الكتالوج الخاص بها ، والكتالوج الخاص بـشرائح المعالج يحتوى عادة على جزأين : **الجزء الأول** يكون خاصا بالتركيب الوظيفي لجميع أجزاء الشريحة ووظيفة جميع أطرافها وشكل الإشارة الناتجة أو المطلوبة على كل طرف من هذه الأطراف . **الجزء الثاني** يحتوى مجموعة أوامر الشريحة والتي بها يمكنك برمجتها وعدد هذه الأوامر يختلف بالطبع من شريحة لأخرى وكذلك صيغ الأوامر تختلف باختلاف الشريحة . لذلك سنحاول من خلال فصول هذا الكتاب أن نقوم بتغطية هذين الجزأين بالإضافة إلى كيفية مواجهة أو توصيل شريحة المعالج على الأجهزة أو الشريحة الخارجية للحصول على نظام الميكروكمبيوتر ذي الكارت الواحد One board microcomputer الذي يستخدم في الكثير من أغراض التحكم وفي الكثير من التطبيقات العملية ، كل ذلك من خلال دراسة مفصلة ومتأنية للمعالجات ذات 8 بت والمعالجات ذات 16 . بعد ذلك سيكون هناك عرضًا سريعاً للمعالجات 32 بت والأجيال الأخيرة منها ودراسة الفرق بينها وبين الأجيال السابقة . قبل أن نترك المقدمة ومحاولة لإتمام النظرة العامة على الموضوع سنعرض في الجزء القادم لتركيب الميكروكمبيوتر وما يمثله المعالج بداخله .

2-1 أين يقع المعالج في داخل الميكروكمبيوتر ؟ (Computer Architecture)

إن الفكرة التي يقوم عليها الحاسب ما هي إلا تقليداً لطريقة الإنسان في حل أي مسألة . أنت مثلاً حينما تريد أن تحل مسألة في الطبيعة أو الإلكترونيات ، ماذا تحتاج ؟ إنك لكي تحل هذه المسألة ستحتاج للآتي :

1. القوانين المهمة لـحل هذه المسألة وبالطبع فإنك سـتستعين بأحد الكتب التي تحتوى على هذه القوانين .

2. سـتحتاج كراسة أو بعض الأوراق لـتدوين بعض النتائج الحسابية .
3. سـتحتاج أيضاً إلى آلة حاسبة لـمساعدتك في إجراء بعض العمليات الحسابية .
4. أخيراً ربما تحتاج إلى آلة طابعة لـكتابة تقرير أو وضع الحل في صورة نهائية لـلائحة .

هذه الأشياء الأربع لو تم توفيرها مجتمعة لن تـحل المسألة من تلقاء نفسها ولكن لابد من وجود منظم لعملية الحل وهو الشخص نفسه ولابد من وجود خطة للحل أيضاً . هذه الأجزاء الأربع السابقة هي تقريباً ما يتـركب منه الحاسب كما سنرى سـوى أن خطة الحل وهي البرنامج يتم وضعها للحـاسـب عن طريق الإنسان بحيث إذا جاء الحل خطـأ فـلابـد أن هناك خطـأ في البرنامج الموضوع للـحـاسـب

بواسطة الشخص المبرمج . أى أن الحاسوب لا يحل المسألة من تلقاء نفسه ولكنه يسير على خطوة الحل التي وضعتها أنت له مستفيداً فقط بالسرعة الهائلة التي ينفذ بها العمليات . لذلك سنعرض الآن للأجزاء الرئيسية في الحاسوب تاركين للقارئ أمر مقارنة هذه الأجزاء بالأجزاء الأربع التي ذكرناها سابقاً . يتكون الميكروكمبيوتر كما هو موضح في شكل (1-1) من ثلاثة أجزاء رئيسية هي :

1. الذاكرة Memory
2. وحدات الإدخال والإخراج Input/Output Ports
3. وحدة المعالجة المركزية Central Processing Unit

1-2-1 الذاكرة Memory

الذاكرة ما هي إلا وعاء لحفظ المعلومات لحين الحاجة إليها وهذه المعلومات إما أن تكون بيانات ستكون هناك حاجة إليها فيما بعد أو تكون برنامجاً مخزنًا في الذاكرة في انتظار التنفيذ . إن أي برنامج تكتبه على الحاسوب وبأى لغة ولتكن لغة الباسيك BASIC مثلاً لابد وأن يوجد أولًا في الذاكرة الأساسية للحاسوب ثم يتم استدعاؤه من هناك للتنفيذ عند الأمر بذلك ، أى أن الحاسوب لا ينفذ إلا ببرامج موجودة في الذاكرة الأساسية فقط . تنقسم الذاكرة عامة إلى قسمين :

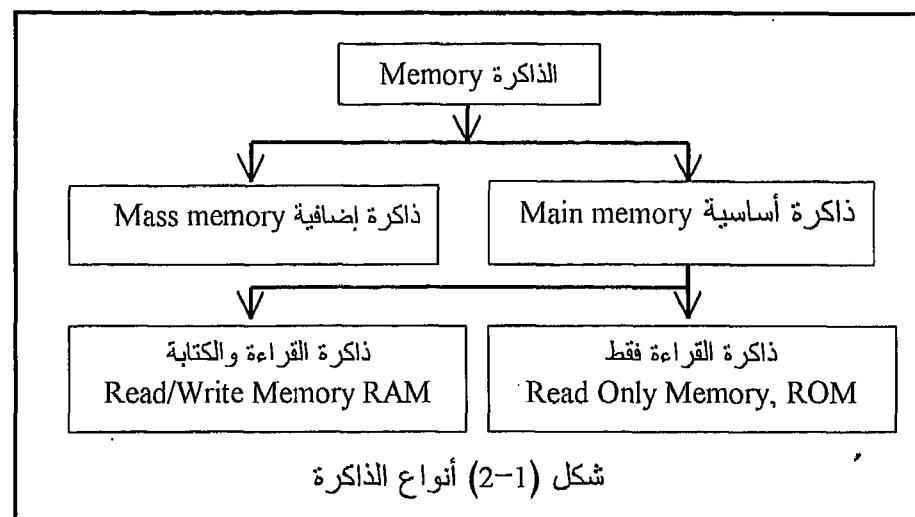
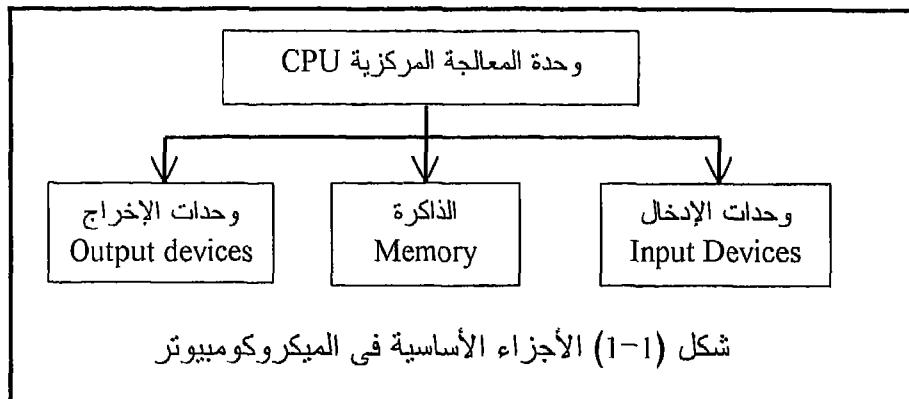
القسم الأول وهو الذاكرة الأساسية للحاسوب Main memory وهي التي تخزن فيها البرامج التي تنتظر التنفيذ ، وهذا النوع من الذاكرة يكون عادة من أشباه الموصلات Semiconductors وعلى شرائح تكاملية ويحدد مقدارها على حسب نوع المعالج المستخدم في الحاسوب كما سنرى فيما بعد . الذاكرة الأساسية للحاسوب تنقسم بدورها إلى جزأين :

الأول : وهو ما يسمى بذاكرة القراءة فقط Read Only Memory , ROM وهذه الذاكرة أيضاً تكون عبارة عن شرائح من أشباه الموصلات التي تم تسجيل محتوياتها بواسطة الصانع نفسه وعادة تحتوى الثوابت والبرامج المهمة لتشغيل نظام الحاسوب والتي لا تضيع بانقطاع مصدر الطاقة عنها .

الثاني : وهو ذكرة القراءة والكتابة Read/Write Memory وقد تم العرف على تسمية هذا النوع من الذاكرة بذاكرة الاتصال العشوائي Random Access Memory, RAM وهي الذاكرة التي تحتوى على البيانات والبرامج التي في انتظار التنفيذ كما ذكرنا من قبل وهذه الذاكرة تفقد محتوياتها بانقطاع مصدر الطاقة .

القسم الثاني من أقسام الذاكرة هو الذاكرة الإضافية أو Mass Memory وهي الذاكرة التي تستخدم لتخزين البيانات أو البرامج لفترات طويلة وعادة فإن هذه الذاكرة تكون مغناطيسية مثل الأقراص Floppy disks والشرائط Tapes وهناك

أيضا الأقراص الصلبة Hard Disks . هذا القسم من الذاكرة لا دخل للمعالج فى تحديد كميته ولكن كميته تحدد على حسب رغبة المستخدم وما وصلت إليه التكنولوجيا في هذا المجال . شكل (1-2) يبين رسميا توضيحيا لأقسام الذاكرة في الحاسوب التي سبق الحديث عنها .



١-٢-٢ وحدات الإدخال والإخراج Input/output Ports

وحدات الإدخال هي الوسائل التي يتم بها تكيف المعلومات لتكون في صورة مناسبة يستطيع البروسيسور التعامل معها ، ومثال ذلك لوحة المفاتيح التي تحول أي زرار تقوم بضغطه إلى إشارات كهربائية وشفرات يقبلها المعالج . يجب أن نفرق هنا بين بوابة الإدخال ووحدة الإدخال حيث بوابة الإدخال يتم من خلالها

إدخال المعلومات التي تم تجهيزها بواسطة وحدة الإدخال إلى المعالج كما سنرى بالتفصيل في فصول الكتاب القادمة .

وأما وحدات الإخراج فهي الوسائل التي يتم بها إظهار المعلومات الخارجة من المعالج ، ومثل ذلك الشاشة التي ما هي إلا وسيلة ضوئية لإظهار المعلومات التي تخرج من المعالج ، بالطبع فإن هذه الشاشة تكون متصلة بأحد بوابات الإخراج ، ولذلك يجب أن نفرق هنا بين بوابات الإخراج ووحدات الإخراج حيث بوابة الإخراج يتم من خلالها إخراج المعلومة من البروسيسور إلى وحدة الإخراج التي تعامل مع هذه المعلومات بوسائل مختلفة كما سنرى بالتفصيل .

1-2-3 وحدة المعالجة المركزية

Central Processing Unit, cpu

الوظيفة الأساسية لوحدة المعالجة المركزية هي تنفيذ البرامج عن طريق إحضار الأوامر من الذاكرة الواحد بعد الآخر ثم تنفيذها بنفس التتابع ، فمثلاً يتم إحضار الأمر الأول ثم ينفذ وبعد ذلك يحضر الأمر الثاني وينفذ فالأمر الثالث وينفذ وهكذا إلى أن تصل إلى نهاية البرنامج . بعض هذه الأوامر تحتاج لبيانات من أماكن أخرى في الذاكرة يتم إحضارها ، وبعضها يحتاج لبيانات من بوابات إدخال يتم إحضارها أيضاً ، والبعض الآخر من الأوامر يتطلب كتابة أو تسجيل بعض البيانات إما في الذاكرة أو في وحدات إخراج ، كل ذلك وأكثر تقوم به وحدة المعالجة المركزية . في معظم أنظمة الميكروكمبيوتر الشخصية تكون وحدة المعالجة المركزية هي شريحة أو أكثر من شرائح الميكروبروسيسور أو المعالج الدقيق كما سمي بالعربية والذي هو موضوع دراسة هذا الكتاب .

1-3 ماذا تعنى هذه الألفاظ ؟

نسمع هذه الأيام الكثير من الألفاظ والتي لا نعرف مدلولها الدقيق ولا ماذا تعنى هذه الألفاظ ؟ لذلك سنقدم في هذا الجزء بعض هذه الألفاظ مع شرح بسيط لمدلولها والاستعانة ببعض الأمثلة إن أمكن .

1-3-1 الميكروكمبيوتر والميكروبروسيسور

لقد رأينا في هذا الفصل كيف أن كلمة ميكروبروسيسور تعني تلك الشريحة ذات الأطراف المتعددة والقادرة على تنفيذ مجموعة من الأوامر المحددة بحيث ينفذ كل أمر عند اعطاء الشفرة الخاصة به . كلما تعددت هذه الأوامر ، وكلما كان المعالج أسرع في تنفيذ هذه الأوامر ، وكلما كان المعالج أسهل في عمليات

المواجهة مع الدوائر المحيطة كلما كان المعالج أفضل . في الكثير من الأحيان يستخدم لفظ "بروسيسور" فقط للدلالة على نفس الشيء ، ونحن في هذا الكتاب سنستخدم أى واحد من اللفظين "بروسيسور" أو "ميكروبروسيسور" أو المرادف العربي لهما وهي كلمة "المعالج" نظراً لشيوخ استخدام كل هذه الألفاظ .

أما الميكروكمبيوتر فقد رأينا سابقاً أنه ذلك الجهاز الذي يتكون من بعض الأجزاء الثانوية مثل الذاكرة ووحدات الإدخال والإخراج وجزء أساسي وهو المعالج . أى أن المعالج يعتبر جزءاً أساسياً بل هو أهم جزء في الميكروكمبيوتر . عند ذكر كلمة ميكروكمبيوتر يتادر إلى ذهننا فوراً تلك المجموعة المكونة من شاشة للعرض ولوحة مفاتيح وطابعة وغير ذلك من الأجهزة ، ولكن في الحقيقة فإن هذا هو أحد أشكال الميكروكمبيوتر موضوعاً في صورة تسهل عملية التعامل معه ويرجعه حتى من غير المختصين الذين يتعاملون معه بغض البرمجة فقط باستخدام اللغات المعروفة . هناك صوراً أخرى للميكروكمبيوتر غير هذه الصورة المألوفة مثل "الميكروكمبيوتر ذو الكارت الواحد" مثلاً وهو عبارة عن كارت واحد عليه شريحة المعالج وشريحة ذاكرة والقليل من بوابات الإدخال والإخراج ، كل ذلك مبني على كارت واحد لأداء غرض معين مثل التحكم في أى عملية صناعية كما سنرى في هذا الكتاب . بل إن هناك صورة أخرى للميكروكمبيوتر وهي الميكروكمبيوتر على شريحة واحدة ، نعم شريحة واحدة تحتوى معالج وبعض الذاكرة (ROM و RAM) وبعض بوابات الإدخال والإخراج . بل إن هناك بعض شرائح الميكروكمبيوتر التي تحتوى الأكثر من ذلك مثل المحولات من تماثل إلى رقمي (A/D) والمحولات من رقمي إلى تماثل (D/A) والمؤقتات (Timers) والمرشحات الرقمية (Digital Filters) وغير ذلك وعادة ما يطلق على هذه الشرائح الحاكمات الدقيقة . Microcontrollers

1-3-2 البرمجة والبناء Software and Hardware

يكون التعامل مع المعالج في العادة بوسيلة من الشتتين لا غنى لواحدة منها عن الأخرى :

الوسيلة الأولى : هي برمجة المعالج وهو ما يسمى software وعادة ما تكون البرمجة بلغة الماكينة الخاصة بالبروسيسور الذي تتعامل معه حيث أن كل بروسيسور له لغة ماكينة خاصة به كما سنرى في هذا الكتاب .

الوسيلة الثانية: هي البناء hardware وتشتمل على مواجهة أو توصيل البروسيسور على الدوائر المحيطة مثل الذاكرة وبوابات الإدخال والإخراج واستخدام البروسيسور في التطبيقات المختلفة مثل دوائر التحكم مثلاً . إن المتعامل مع المعالج لا بد وأن يكون ملماً بكلتا الوسائلتين السابقتين ، البرمجة

والبناء ، وإن اختلفت نسبة المame بـأى واحدة منها ، فإنه من الصعب لشخص ما أن يتعامل مع البروسيسور بعرض البرمجة فقط لأن ذلك سيعتبر إهاراً لوقته حيث إنه إذا كان يريد عرض البرمجة فقط فيمكنه استخدام أى واحدة من اللغات المعروفة والتي يكون تعلمها أسهل بكثير من تعلم برمجة البروسيسور بلغة الماكينة أو الأسمايلى . أى أن من يتعامل مع البروسيسور فإنه غالباً يتعامل معه بغرض استخدامه كعنصر فعال في دائرة تحكم لمميزاته المختلفة ، لذلك لا بد لهذا الشخص أن يتعلم برمجة البروسيسور وواجهته واستخدامه فى الأغراض المختلفة . وعادة تطلق كلمة Software على البرمجة بأى لغة من اللغات وأما كلمة Hardware فتطلق على عمليات البناء والتركيب الإلكتروني بأى شكل ولأى غرض .

1-3-3 الأمر والبرنامـج Instruction and Program

- أعطنى القلم ! ! ! ! ! هذا أمر Instruction وأما
1. خذ المفتاح
 2. افتح درج المكتب
 3. خذ القلم
 4. اكتب "أنا أتعلم البرمجة"
 5. أعد القلم
 - 6.أغلق الدرج
 7. أعطنى المفتاح

!!!!!!!!!!!!!!

فهذا برنامج مكون من سبعة أوامر .

أى أن الأمر هو وحدة بناء البرنامج ، بينما البرنامج يتكون من مجموعة من الأوامر التي لو تم تنفيذها بالتتابع المحدد فإنها تؤدى هدف أو وظيفة معينة . أى برنامج لا بد وأن ينفذ بالتتابع الذى كتب به وهذا هو ما يفعله المعالج بحيث إذا وجد هناك خطأ ما في النتيجة فإن ذلك يرجع إلى البرنامج وليس إلى المعالج لأن المعالج ما هو إلا ماكينة مطيعة تنفذ ما يطلب منها حتى ولو كان خطأ طالما أن ذلك في حدود إمكانياتها .

1-3-4 لغات البرمـحة Programming Languages

تخيل أن لدينا ماكينة وهذه الماكينة لها 3 مفاتيح تشغيل ، كل مفتاح يكون إما ON أو OFF ، وعلى ضوء حالة المفاتيح الثلاثة يمكن لهذه الماكينة أن تنفذ عملية معينة . جدول 1-1 يبين جميع الحالات الممكنة للمفاتيح الثلاثة والعملية المقابلة لكل حالة . هذه العمليات التي تستطيع الماكينة تنفيذها هي مجموعة

الأوامر الخاصة بهذه الماكينة وعلى ضوء هذه العمليات وبالاستعانة بالجدول 1-1 فإننا نستطيع إعادة كتابة البرنامج الموجود في الجزء السابق في صورة شفرات تستطيع هذه الماكينة تنفيذها ، جدول 1-2 يبين هذا البرنامج وقد أعيد كتابته وفي مقابل كل أمر الشفرة الخاصة به . مجموعة الأوامر المكتوبة بالوحaid وأصفار في جدول 1-2 هي البرنامج مكتوبا بلغة الماكينة لهذه الآلة التي نستخدمها . لذلك فإنه عامة وكما سنرى في هذا الكتاب فإن لغة الماكينة تكون عبارة عن شفرات ثنائية (وحaid وأصفار) تدخل إلى بروسيسور فينفذها وهي الصورة الوحيدة التي يمكن التعامل بها مع أي بروسيسور ، وكما سنرى أيضا فإن لكل معالج لغة الماكينة الخاصة به .

S2 S1 S0	العملية المنفذة
0 0 0	خذ (يأخذ أي شيء يعطى له في يده)
0 0 1	افتح (يستخدم ما معه لفتح ما يحدد له)
0 1 0	أكتب (يستخدم ما معه للكتابة)
0 1 1	ضع (ضع ما معك في المكان المحدد)
1 0 0	أغلق (يغلق ما يحدد له)
1 0 1	أعطي (يعطى ما معه)
1 1 0	اذهب (يذهب إلى مكان محدد)
1 1 1	تكرار (يكسر ما يحدد له عدد من المرات)

جدول 1-1 مجموعة أوامر ماكينة افتراضية

مما سبق يتضح لنا أن الكتابة بلغة الماكينة ليست سهلة وأصعب منها اكتشاف الأخطاء فيها ، لذلك فإنه يمكن الكتابة بالأوامر خذ وافتتح واكتب وهكذا ولكن في هذه الحالة بدلا من إدخال الأوامر على الماكينة مباشرة فإننا ندخلها على مترجم يقوم بترجمة الكلمات خذ واكتب إلى شفراتها الثنائية ثم يدخلها على الماكينة . هذا المترجم يسمى أسمبلر والأوامر المكتوبة بلغة أكتب وخذ سنسميهما لغة الأسمبل . هذا هو الوضع تماما في حالة المعالج حيث أن كل بروسيسور له لغة ماكينة وأسمبلر ولغة أسمبل خاصة به . لذلك فإن لغة الأسمبل هي أقرب اللغات من مستوى لغة الماكينة . الآن ما رأيك لو استطعنا تصميم مترجم آخر يأخذ أوامر أكثر تعقيدا مثل "افتح الدرج وخذ القلم وأكتب" حيث سيقوم المترجم بترجمة ذلك الأمر إلى عدد من أوامر لغة الماكينة وليس إلى أمر واحد كما في حالة الأسمبل ، مثل هذه اللغة التي يكون كل أمر فيها يؤدى وظيفة أكثر من أمر من أوامر لغة الأسمبل تسمى باللغات ذات المستوى العالى High level

language ومنها على سبيل المثال لا الحصر لغة الباسيك والفورتران والبسكال وغيرها كثير . إن المترجم الذى يقوم بالترجمة من لغة ذات مستوى عال إلى لغة ماكينة يسمى المؤلف أو المفسر أو المترجم أو compiler .

الأمر	حالة المفاتيح	S2 S1 S0
خذ (المفتاح)		0 0 0
افتح (درج المكتب)		0 0 1
خذ (القلم)		0 0 0
اكتب (أنا أتعلم البرمجة)		0 1 0
ضع (القلم)		0 1 1
أغلق (الدرج)		1 0 0
أعطني (المفتاح)		1 0 1

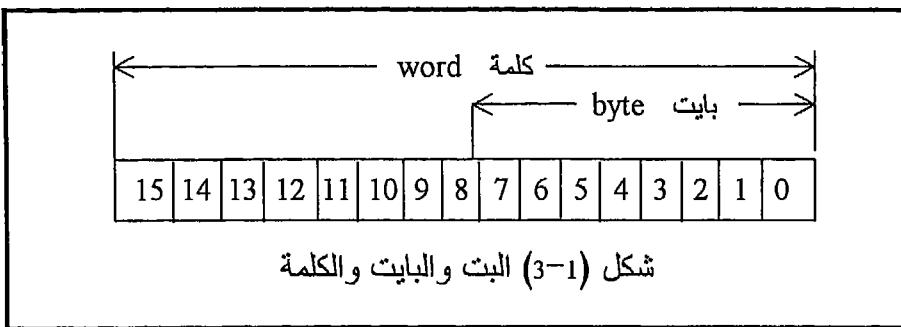
جدول 1-2 برنامج مكتوب بلغة الماكينة الافتراضية

5-3-1 البٰٰيت bit والبٰٰيٰات byte

البٰٰيت هى الخانة فى نظام العد الثنائى ، فكما أن العدد العشري 325 مثلاً مكون من ثلاثة خانات فإن العدد الثنائى 11001 مكون من خمس خانات أو خمس بٰٰيٰات bits حيث كل خانة تحتوى على واحد أو صفر . عملياً وكما نعلم من الإلكترونيات الرقمية فإن البٰٰيت تكون عبارة عن قلاب flip flop أو أحياناً تسمى ماسك latch يتم وضع خرجه على القيمة واحد أو صفر . كل ثمانية بٰٰيت تكون فيما بينها ما يسمى بالبٰٰيٰات byte ، والبٰٰيٰات هي وحدة تدبير الذاكرة فنقول مثلاً أن هذا الحاسوب ملحق به ذاكرة مقدارها 64 كيلو بٰٰيت أى 65536 بٰٰيٰات حيث أن الواحد كيلو بٰٰيت يساوى 1024 بٰٰيت كما سنعرف بالتفصيل فيما بعد . شكل (1-3) يبين الفرق بين البٰٰيت والبٰٰيٰات . كل اثنين بٰٰيت تكونان ما يسمى بالكلمة word وعلى ذلك فإن الكلمة word تتكون من 16 بت أو اثنين بٰٰيت كما هو موضح فى شكل (1-3) أيضاً .

4-1 تمارين

1. وضح بالرسم الصندوقى أجزاء الميكروكمبيوتر وشرح باختصار وظيفة كل جزء ؟



2. طابق بين الأجزاء التي شرحتها في السؤال السابق وما تحتاج إليه من أشياء لحل مسألة في الرياضيات مثلاً كما هو موضح في الفصل ؟
3. اشرح أنواع الذاكرة وخصائص كل نوع ؟
4. دليل التليفون ، هل تطابقه مع RAM أم ROM ؟
5. شريط الكاسيت ، هل تطابقه مع RAM أم ROM ؟
6. القرص الممعنط floppy disk ، هل هو RAM أم ROM ؟ وإذا صنفته على أنه RAM فهل هي أساسية أم إضافية ؟
7. الفأرة mouse ، هل هي وحدة إدخال أم وحدة إخراج ؟
8. الطابعة ، هل هي وحدة إدخال أم وحدة إخراج ؟
9. المساح scanner ، هل هو وحدة إدخال أم وحدة إخراج ؟
10. الراسم plotter ، هل هو وحدة إدخال أم وحدة إخراج ؟
11. بافتراض أن لك تعاملات مسبقة مع الحاسوب الآلي ، فما نوع المعالج الموجود في الحاسوب الذي تتعامل معه ؟
12. ما هو الفرق بين المعالج والميكروكمبيوتر ؟
13. إذا شبها الميكروكمبيوتر بالسيارة ، فماذا يمثل المعالج في هذه السيارة ؟
14. ارسم مخطط سير flow chart لنشاطك اليومي من الصباح حتى النوم في أيام العمل وأيام العطلات ؟

الفصل الثاني

البناء المعماري للمعالج

Microprocessor Architecture

2-1 مقدمة

في هذا الفصل سيتم عرض المهام الأساسية المطلوبة من أي معالج بصفة عامة وعلى ضوء هذه المهام سنعرض الوظائف الأساسية للمكونات العامة لأى شريحة معالج ، ثم نقدم التركيب التفصيلي لاثنين من الشرائح المعروفة وهى الشريحة Intel8085 والشريحة Z80 على أساس أن هذه هي أكثر شرائح الجيل الثاني استخداما وبعد ذلك سنترك للقارئ رؤية مدى ملائمة هذا التركيب للمهام المطروحة . وسوف نعرض التركيب التفصيلي لهذين المعالجين بصورة مختصرة وسريعة حتى يمكن أي قارئ من مراجعة المكونات الأساسية لهما حتى ولو كان لا ينوى التدريب إلا على أحدهما . لذلك فإننا ننصح بقراءة هذا الفصل بأكمله من مستخدمي هذين المعالجين بالذات أو المعالجات التي ترى نقوم بعرضها في هذا الكتاب . ولقد رأينا في الفصل السابق (عصر المعالجات) أن وظيفة المعالج الأساسية هي إحضار الأوامر من الذاكرة وتفيذها الواحد بعد الآخر ، ولذلك فإن تركيبة الداخلي يجب أن يناسب هذه المهمة أو الوظيفة .

2-2 المهام الأساسية المطلوبة من المعالج

1. يجب أن يكون المعالج قادرًا على إحضار معلومات من الذاكرة (هذه المعلومات قد تكون بيانات يحتاجها في عملية تنفيذ الأوامر أو قد تكون الأوامر نفسها) .
2. يجب أن يحتوي المعالج على مكان مناسب بداخله لحفظ هذه المعلومات التي أحضرها لحين الحاجة إليها أو تفيذه إذا كانت من الأوامر .
3. لابد أن يكون هناك أكثر من مكان بداخله بحيث يمكن نقل المعلومات فيما بين هذه الأماكن حيث تحتاج بعض الأوامر لذلك عند تفيذهما .
4. يجب أن تكون لديه الوسائل المناسبة لإدخال معلومات من بوابات إدخال حتى يتسعى لنا قراءة لوحة مفاتيح أو إدخال درجة حرارة مثلاً تمهدًا لمعالجتها رقمياً .
5. يجب أن تكون لديه المقدرة على إجراء بعض العمليات الحسابية والمنطقية على البيانات التي أحضرها . العمليات الحسابية الأساسية هي الجمع والطرح والعمليات المنطقية الأساسية مثل AND و OR و NOT .
6. المقدرة على إرسال بيانات إلى الذاكرة وتسجيلها فيها من المهام الأساسية للمعالج .

7. المقدرة على إرسال بيانات إلى وحدات إخراج من خلال بوابات إخراج حتى يتسعى لنا قراءة هذه المعلومات على شاشة أو إخراج بيانات نتحكم بها في سرعة موتور مثلاً .

كانت هذه هي المهام الأساسية للمعالج والتي يجب أن يتحققها تركيبه الداخلى ومجموعة أوامره كما سنرى . سنبدأ فيما يلى الحديث عن مجموعة المسجلات والعدادات التي يشتمل عليها أي معالج حيث أنه من الضروري لأى مستخدم للمعالج أن يعرف خصائص تلك المسجلات ووظائفها .

2-3 أجزاء المعالج الأساسية

جميع شرائح المعالجات تتربّك من ثلاثة أجزاء رئيسية وهي :

1. مجموعة مسجلات وعدادات .
2. وحدة الحساب والمنطق ALU .
3. وحدة التزامن Clock .

بالنسبة لوحدة الحساب والمنطق سوف نرجئ الحديث عنها الآن حيث سيتم إفراد فصل قادم خاص بها (الفصل السادس) وأما وحدة التزامن فسوف يتم الحديث عنها أيضاً لاحقاً وفي معرض الكلام عن وظيفة كل طرف من أطراف شريحة المعالج . أما مجموعة المسجلات والعدادات ووظيفة كل منها فسوف تكون الموضوع الأساسي في الجزء القادم .

2-4 المسجلات والعدادات في شريحة المعالج

تستخدم المسجلات للتخلّي المؤقت للمعلومات في صورة خانات ثنائية في داخل شريحة المعالج لحين الحاجة إليها . إن أي مسجل إزاحة يمكن تصميمه ليكون قادرًا على أداء الوظائف التالية :

1. إدخال المعلومات بالتوالي وإخراجها بالتوالي (سواء من الشمال لليمين أو من اليمين للشمال) .
2. دوران المعلومات في أي اتجاه وعكسه .
3. إدخال المعلومات بالتوازي وإخراجها بالتوازي .
4. إدخال المعلومات توالياً من أي اتجاه وإخراجها توازي أو العكس .

المسجلات داخل المعالج يمكن النظر إليها على أنها واحد من نوعين ، الأول هو مسجلات عامة الأغراض general purpose registers وهذه تستخدم في الكثير من الأغراض وتؤدي أكثر من وظيفة واحدة تكون هذه المسجلات متاحة

للمستخدم لكي يتعامل معها ، إما أن يسجل فيها أو يقرأ منها ، وأما النوع الثاني فهو مسجلات خاصة الأغراض dedicated registers وهذه مسجلات موجودة لأداء غرض أو وظيفة واحدة لا تحد عنها وليس للمستخدم أى وسيلة للتحكم فيها سواء بالقراءة منها أو الكتابة فيها .

أما العدادات counters فتستخدم عادة لعد النبضات الداخلة إليها ويمكن توظيف هذه العدادات لكي تقوم بعملية العد إما تصاعدياً أو تناظرياً مع ملاحظة أن خرج العدادات يكون دائماً توازي . سنعرض فيما يأتي بشكل عام لوظيفة كل مسجل من المسجلات الرئيسية لشريحة المعالج وذلك دون تخصيص معالج معين لأن ذلك مطبق على جميع المعالجات التي سنتعامل معها في هذا الكتاب .

2-4-1 مسجل التراكم A Accumulator

أي مسجل يمكن النظر إليه على أنه بait من بaitات الذاكرة وهذه البايت موجودة داخل شريحة المعالج وعادة تكون هناك حرية أكثر في التعامل مع البيانات الموجودة داخل هذه المسجلات عن البيانات الموجودة في الذاكرة . من هذه المسجلات ما يسمى بـ مسجل التراكم Accumulator أو المركم . يعتبر مسجل التراكم ، وعادة يرمز له بالرمز A ، من أكثر مسجلات المعالج عملاً ولذلك فإنه يمكننا النظر إليه على أنه سكريپترا لشريحة المعالج . إن أي عملية حسابية أو منطقية يقوم بها المعالج لابد وأن يكون مسجل التراكم طرفاً فيها ، فمثلاً لو أردت أن تجمع أي رقمين فإن واحداً منها لابد وأن يوضع في مسجل التراكم وأما الرقم الآخر فيوضع في أي مسجل آخر أو حتى في الذاكرة ، ليس هذا فقط بل إن نتيجة أي عملية حسابية أو منطقية لا توضع إلا في مسجل التراكم ومنه يمكن نقلها لأي مكان آخر وذلك في المعالجات 8 بت . هناك مهمة أخرى أيضاً لهذا المسجل وهي أن أي عملية إدخال أو إخراج من خلال بوابات الإدخال أو الإخراج عادة تكون من خلال هذا المسجل . أي أن المعلومة توضع في مسجل التراكم أولًا ثم يتم إخراجها إلى بوابة الإخراج ، أو إذا كانت المعلومة قادمة من بوابة إدخال فإنها توضع أولًا في مسجل التراكم ثم يتم نقلها منه لأي مكان آخر في داخل البروسيسور أو خارجه .

إذن ما رأيك الآن في تسميته بـ مسجل التراكم ؟ إن عدد البaitات (الخانات) bits الموجودة في مسجل التراكم دائماً يساوى عدد خطوط مسار البيانات data bus ومن الممكن في بعض المعالجات أن يكون هناك أكثر من مسجل تراكم واحد كما سنرى . بعض هذه الوظائف الخاصة بالمركم سيتم الاستغناء عنها في المعالجات 16 بت كما سنرى .

2-4-2 عدد البرنامج Counter, PC

كما علمنا فإن مهمة المعالج الأساسية هي إحضار الأوامر من الذاكرة الواحد بعد الآخر ثم تفيدها ، ولذلك فإنه لابد لهذه المهمة من تحديد للأماكن التي تحتوى هذه الأمر في الذاكرة . يحتوى عدد البرنامج دائمًا على عنوان المكان في الذاكرة الذي يحتوى الأمر الذي عليه الدور في التنفيذ ، وكلما تم إحضار أي أمر من الذاكرة وقبل أن يتم تفيذه فإن عدد البرنامج تتغير محتوياته بحيث تشير إلى عنوان الأمر القادم في التنفيذ . تذكر أيضًا أنه حتى لو حدث قفز من مكان في البرنامج إلى مكان آخر فإن وحدة التحكم داخل المعالج تضع عنوان الأمر الذي سيتم القفز إليه في عدد البرنامج حتى يصبح هو الأمر الذي عليه الدور في التنفيذ وتنقل عملية تفيذ البرنامج إلى هناك . عدد برات هذا العدد دائمًا تساوى عدد مسار العنوانين address bus و هذا منطقى جدا حتى يتمكن المعالج من إحضار الأوامر مهما كانت في أي مكان في الذاكرة سواء في أولها أو في آخرها ، لاحظ أن كمية الذاكرة التي يمكن أن يتعامل معها المعالج تتوقف على عدد البتات أو الخطوط في مسار العنوانين كما سنرى فيما بعد . إذا نظرنا إلى عدد البرنامج على أنه مسجل يحتوى عنوان الأمر الذي عليه الدور في التنفيذ فإننا سنصنفه على أنه من المسجلات ذات الأغراض الخاصة dedicated register ، لاحظ أيضًا أنك كمبرمج لا تستطيع التحكم في محتويات هذا العدد .

3-4-2 مسجل وفاك شفرة الأوامر

Instruction Register And Decoder

بعد أن يتم إحضار الأمر من الذاكرة إلى شريحة المعالج لابد وأن يسجل أو يوضع في أحد الأماكن في انتظار تفيذه ، هذا المكان هو مسجل الأوامر . أي أن مسجل الأوامر يحتوى شفرة الأمر الذي يتم تفيذه الآن . لاحظ أن عدد برات مسجل الأوامر عادة يساوى عدد برات البايت في الذاكرة التي تساوى بدورها عدد برات مسار البيانات خاصة في هذا الجيل من المعالجات الذى نحن بصدده الآن ، كما أن عدد الأوامر التي يمكن للمعالج أن ينفذها سيتوقف على عدد البتات في مسجل الأوامر فمثلاً إذا كان عدد برات مسجل الأوامر هو 8 بت فإن ذلك يعني أن هذا المعالج يستطيع التعامل مع $2^8 = 256$ أمر على الأكثر .

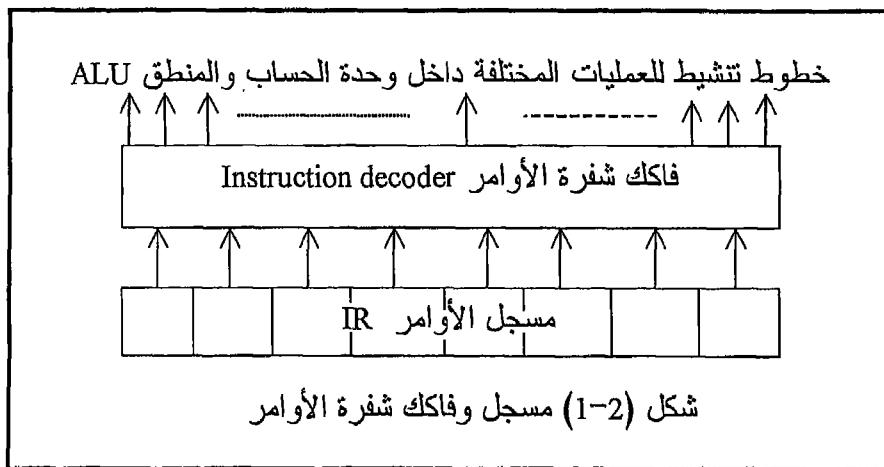
أول خطوات تفيذ أي أمر تبدأ من فاك شفرة الأوامر الذي يتصل دخله بخرج مسجل الأوامر كما في شكل (1-2) بحيث أنه على حسب شفرة الأمر الموجودة في مسجل الأوامر فإن عملية واحدة فقط سيتم تنفيذها على حسب الشفرة الموجودة على دخل فاك الشفرة ويتم ذلك بالطبع بمساعدة وحدة التحكم ووحدة الحساب والمنطق .

4-4-2 مسجل الحالة Status Register, SR

أحياناً يطلق على هذا المسجل اسم مسجل الأعلام Flag Register, FR . يعتبر هذا المسجل نشرة إخبارية تعكس حالة نتيجة آخر عملية حسابية أو منطقية قام المعالج بتنفيذها ، فمن هذا المسجل نستطيع أن نعرف مثلاً إذا كانت هذه النتيجة سالبة أم موجبة أم تساوى صفرًا وغير ذلك من الأخبار المفيدة . هذا المسجل يحتوى على عدد من البتات وكل واحدة منها تعتبر علما flag يعكس أو يدل على حالة معينة من العملية الحسابية أو المنطقية التي تم تنفيذها ، من هذه الأعلام ما يلى :

4-4-2-1 علم الصفر Zero flag, ZF هذه البت تكون واحداً إذا كانت نتائج آخر عملية حسابية أو منطقية تساوى صفرًا وتكون هذه البت صفرًا إذا كانت النتيجة مختلفة عن الصفر سواء موجبة أو سالبة .

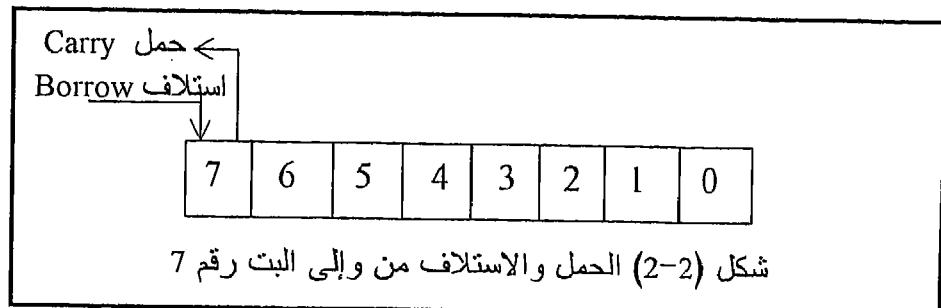
4-4-2-2 علم الإشارة Sign flag, SF هذه البت تكون واحداً إذا كانت نتائج آخر عملية حسابية أو منطقية تساوى صفرًا ، أما إذا كانت هذه النتيجة موجبة فإن هذا العلم يكون صفرًا ، لذلك فإنه أحياناً يسمى بعلم السالبية Negative Flag, NF . لاحظ أن آخر بت في النتيجة تعكس إشارتها فإذا كانت آخر بت تساوى صفرًا فإن ذلك يعني أن النتيجة موجبة أما إذا كانت هذه البت واحداً فإن ذلك يعني أن النتيجة سالبة لذلك فإنه دائماً تكون محتويات علم الإشارة تساوى محتويات آخر بت في النتيجة .



4-4-3 علم الحمل Carry flag, CF هذا العلم يكون واحداً إذا حصل حمل من آخر بت في أي عملية جمع أو حصل استلاف Borrow لآخر بت carry

في أي عملية طرح ويكون صفرًا إذا لم يكن هناك حمل أو استلاف في آخر عملية حسابية . شكل (2-2) يبين كل من عمليات الحمل والاستلاف من وإلى البت رقم 7 .

4-4-4 علم الباريتي Parity flag, PF هذا العلم يكون واحداً إذا كانت آخر عملية حسابية أو منطقية قام بها المعالج تحتوي على عدد زوجي من الوحدات أما إذا كانت هذه النتيجة تحتوي على عدد فردی من الوحدات فإن هذا العلم يكون صفرًا .



4-4-5 علم الحمل النصفي أو الثنائي Half carry flag, HC هذا العلم يكون واحداً إذا كان هناك حمل من الخانة أو البت الثالثة إلى البت الرابعة نتيجة أي عملية جمع أو هناك استلاف من البت الرابعة إلى البت الثالثة نتيجة أي عملية طرح ، ويكون صفرًا فيما عدا ذلك أي إذا لم يحدث استلاف أو حمل من أو إلى البت الرابعة ، لاحظ أننا هنا نبدأ عملية عد البقاعات بالرقم صفر ، أي أن أول بت هي البت رقم صفر . شكل (2-3) يبين كيفية تأثير علم الحمل النصفي . التطبيق على جميع الأعلام واستخدامها سيأتي عند الشرح التفصيلي لأوامر المعالج ، مع العلم أن عدد الأعلام سيختلف من معالج لآخر كما سنرى عند دراستنا للتراكيب التفصيلي لكل بروسيسور سندرسه في هذا الكتاب ولكن دعنا الآن ننظر للمثال التالي كتطبيق سريع على هذه الأعلام .

مثال 1.2

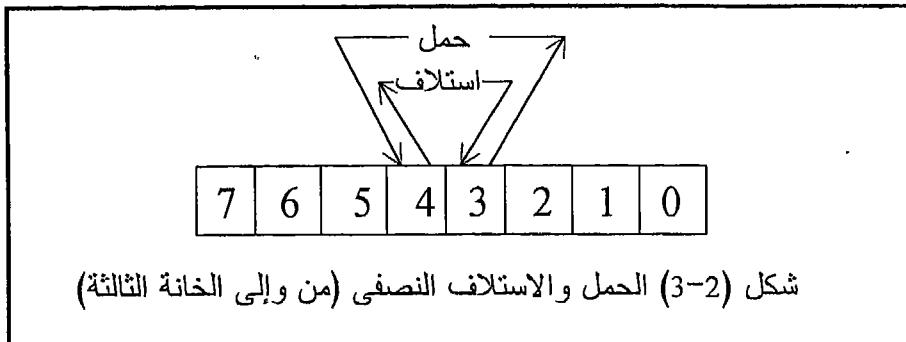
- اكتب محتويات الأعلام السابقة بعد إجراء عملية جمع الرقمين 77H و A5H .
لاحظ أن الرقمين مكتوبين في الصورة المستعشرية hexadecimal .

الجمع الثنائي للرقمين السابقيين سيتم كما يلى :

الرقم الأول	0111 0111
الرقم الثاني	1010 0101
النتيجة	0001 1100 حمل 1

نلاحظ الآتى من النتيجة السابقة :

1. النتيجة لا تساوى الصفر ، إذن فعلم الصفر يساوى صفر $ZF=0$.
2. آخر بت فى النتيجة صفر فالنتيجة موجبة وعلم الإشارة يساوى صفر $SF=0$.
3. هناك حمل من البت السابعة (الأخيرة) فعلم الحمل يساوى واحد $CF=1$.
4. النتيجة تحتوى ثلاثة وحيد (عدد فرد) فعلم الباريتى يساوى صفر $PF=0$.
5. ليس هناك حمل من الخانة الثالثة للرابعة فعلم الحمل النصفي يساوى صفر $HCF=0$.



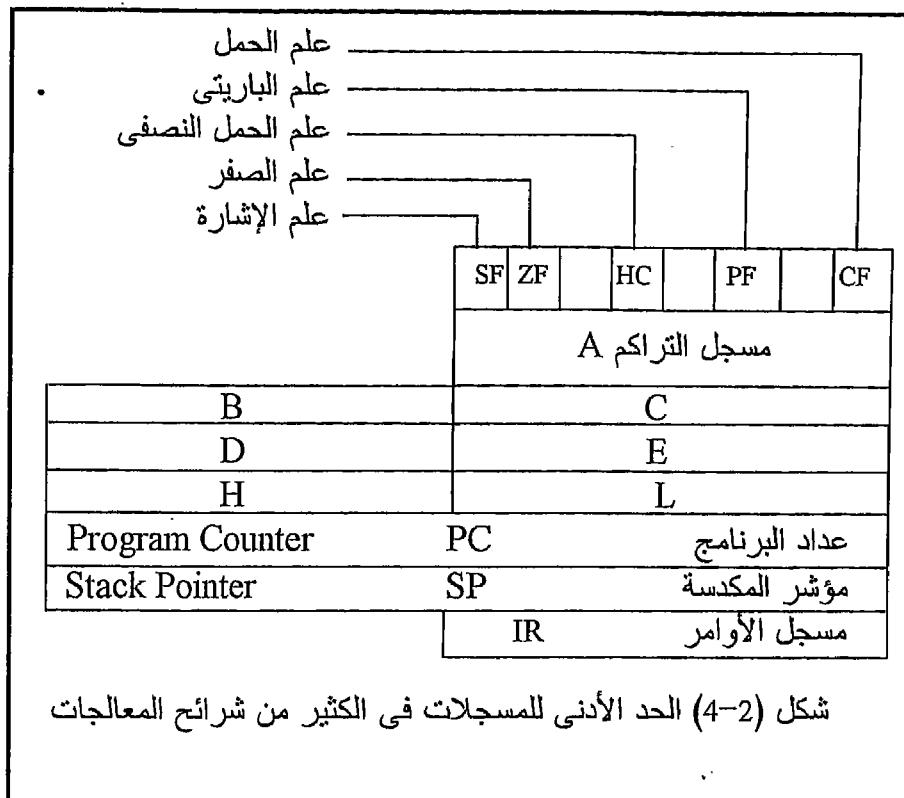
2-4-5 مسجل مؤشر المكذسة Stack Pointer register, SP

سيأتى إن شاء الله شرحًا تفصيلياً للمكذسة stack فيما بعد في معرض الكلام عن البرامج الفرعية وبرامج المقاطعة ، ولكن الآن بإمكانك أن تعرف أن المكذسة هي جزء من الذاكرة يتم فيه تخزين بعض العناوين أو البيانات المهمة والتي لابد من الحاجة إليها واسترجاعها مرة ثانية وبنفس الترتيب الذي تم تخزينها به . مسجل مؤشر المكذسة يحتوى عنوان آخر مكان تم التسجيل فيه في هذا الجزء من الذاكرة ، لذلك فإنه طالما أن هذا المسجل سيحتوى على عنوان فلا بد أن يكون 16 بت . لاحظ أن المبرمج عادة تكون لديه الحرية في اختيار الجزء من الذاكرة الذي سيستعمل كمكذسة .

2-4-6 المسجلات عامة الأغراض General Purpose Registers

في الكثير من الأحوال عندما نجمع أكثر من رقم ، نحتاج لحفظ نتيجة معينة لحين استخدامها في عملية أخرى لاحقة ، ولذلك فإنه بدلاً من إرسال هذه النتيجة إلى الذاكرة ثم استدعائها ثانية مما يأخذ الكثير من الوقت فقد تم تجهيز المعالج ببعض المسجلات التي تستخدم لتخزين مثل هذه النتائج المرحلية لحين الحاجة إليها . عدد البتات في هذه المسجلات يكون عادة متساوية لعدد بิตات مسار البيانات . عدد هذه المسجلات يختلف من معالج لآخر ومن شركة لأخرى . ولقد

تم التعارف على تسمية هذه المسجلات بالمسجلات B و C و D و E و H و L كما سمى المركم من قبل بالمسجل A . هذه التسمية كما سنرى هي التسمية التي ستستخدم مع لغة الأسمبلی (التجميع) assembly language . شكل (4-2) يبيّن جميع المسجلات التي تكلمنا عنها حتى الآن والتي تمثل كما ذكرنا الحد الأدنى لمحطيات أى معالج من المسجلات .



هناك بعض الأوامر التي تتعامل مع هذه المسجلات كأزواج يتكون كل زوج منها من 16 بت بدلاً من التعامل معها كمسجلات يحتوى الواحد فيها على 8 بتات فقط . في هذه الحالة يكون كل مسجل له مسجل آخر يمكن ازدواجه معه ولا يمكن ازدواجه مع أى مسجل آخر ، فمثلاً المسجل B لا يزدوج إلا مع المسجل C فقط وكذلك المسجل D لا يزدوج إلا مع المسجل E والمسجل H لا يزدوج إلا مع المسجل L . لاحظ أنه في حالة ازدواج المسجل B والمسجل C فإن المسجل C يحتوى أو يمثل البایت ذات القيمة الصغرى low significant byte المكونة من 16 بت والمسجل B يحتوى البایت ذات القيمة العظمى high significant byte من هذه المعلومة . بنفس الطريقة في حالة الأزواج DE و HL

فإن المسجلات E و L تحتوى البايت ذات القيمة الصغرى والمسجلات D و H تحتوى البايت ذات القيمة العظمى . فمثلاً إذا أردنا أن نسجل المعلومة 4CF6H المكونة من 16 بت في زوج المسجلات HL فإن البايت F6 وهى البايت ذات القيمة الصغرى لابد أن توضع في المسجل L وأمّا البايت 4C ذات القيمة العظمى فتوضع في المسجل H . في شكل (2-4) ستلاحظ أن هذه المسجلات موضوعة بنفس طريقة وكيفية ازدواجها .

إن التعامل مع هذه المسجلات من خلال المعالج يتم عن طريق شفرة أو كود code تم إعطاؤه لكل واحد من المسجلات العامة وكل زوج منها بحيث يعرف كل مسجل في لغة الماكينة كما سنرى فيما بعد بهذه الشفرة أو هذا الكود . إن هذه الشفرة كما هو موضح في جدول 2-1 مكونة من حايد وأصفار فقط وهذا يتاسب مع متطلبات لغة الماكينة . تذكر أيضاً أن المكونات التي رأيناها إلى الآن ما هي إلا أقل ما يمكن أن يحتويه أي بروسيسور وإن اختلف عددها من بروسيسور لآخر كما سنرى .

الشفرة	السجل
Code	Register
111	A
000	B
001	C
010	D
011	E
100	H
101	L
110	M

Register pairs	أزواج المسجلات
00	BC
01	DE
10	HL
11	SP

جدول 2-1 المسجلات وأزواج المسجلات وشفراتها الثانية

2-5 نظرة خارجية على شرائح المعالج

إن مجموعة شرائح المعالجات ذي 8 بتات التي ندرسها في هذا الكتاب كلها لها عدد 40 طرفاً تخرج منها ، وكذلك المعالج 8086 ذو 16 بت . ابتداء من المعالج 80186 بدأ عدد أطراف هذه المعالجات في الزيادة حيث أصبح 68 طرفاً في

المعالج 80186 ، وظل في الزيادة إلى أن وصل إلى 296 طرفا في حالة المعالج بنتيم برو Pentium Pro وهو آخر المعالجات التي ستدرسها في هذا الكتاب . فما هي وظيفة كل طرف من هذه الأطراف ، ولماذا كل هذا العدد من الأطراف ؟ إننا هنا سنحاول إلقاء نظرة سريعة على وظائف الأطراف الأساسية فقط وسوف نرجي الحديث التفصيلي عنها وشكل الإشارات على كل طرف وكيفية ربط هذه الأطراف بالعالم المحيط بشريحة المعالج إلى فصول خاصة بذلك . هذه الأطراف يمكن تقسيمها إلى المجموعات التالية :

2-5-1 مسار العنوانين Address bus

أى مكان يريد المعالج أن يتعامل معه سواء كان ذاكرة أو غيرها لا بد وأن يحدد المعالج عنواناً لهذا المكان . هذا العنوان يتم وضعه في صورة شفرات كهربائية من الوحدات والأصفار بواسطة المعالج على عدد من هذه الأطراف الخارجة من المعالج تسمى مسار العنوانين . لذلك فإنه على حسب عدد هذه الأطراف المخصصة لحمل شفرة العنوانين يتحدد عدد الأماكن التي يمكن للبروسيسور أن يتعامل معها ، ودائماً يكون عدد هذه الأماكن يساوى 2 مرفوعة لأس عدد هذه الخطوط أو الأطراف . في جميع الشرائح 8 بت والتي نحن بصدد الكلام عنها يكون عدد أطراف مسار العنوانين يساوى 16 طرفاً لذلك فإن مقدار الذاكرة التي يتعامل معها مثل هذا البروسيسور يساوى $2^{16} = 65536$ بait = 64 كيلوبايت باعتبار أن كل واحد كيلوبايت يساوى 1024 بait . لاحظ أن الإشارة الموجودة على مسار العنوانين تكون دائماً خارجة من البروسيسور إلى الأجهزة الخارجية وليس العكس لأن البروسيسور هو فقط الذي يحدد العنوان الذي يريد التعامل معه . جدول 2-2 يبين علاقة بين عدد خطوط مسار العنوانين وكمية الذاكرة التي يمكن التعامل معها في كل حالة .

2-5-2 مسار البيانات Data bus

بمجرد أن يحدد المعالج المكان الذي يريد التعامل معه عن طريق العنوان الذي وضعه على مسار العنوانين يقوم المعالج بإخراج أو استقبال المعلومة نفسها على أو من مسار آخر وهو مسار البيانات . هذا المسار أيضاً عبارة عن عدد من الخطوط تصل بين المعالج والأجهزة المحيطة حيث تسير عليها البيانات المطلوب تداولها بين المعالج والأجهزة خارجه . إن عدد البتات التي تعرف بها أي شريحة معالج يكون على حسب عدد بิตات أو أطراف مسار البيانات ، فالشريحة 8 بت سميت كذلك لأن لها مسار بيانات مقداره 8 بت والشريحة 16 بت سميت كذلك لأن لها مسار بيانات 16 بتاً وهكذا . في عالم الحاسوب توضع أي معلومة دائماً في صورة عدد من الخانات أو البتات وكل بت أو خانة من هذه

الخانات يوضع بها واحد أو صفر حيث يمثل الواحد بجهد معين ويمثل الصفر بجهد آخر . التركيبة المكونة من هذه الوحدات والأصفار هي ما يسمى بالشفرة الثنائية للمعلومة . إن ثمانية من هذه البتات أو الخانات تسمى بآيت واثنين بآيت تسمى كلمة أو Word . كمثال على هذه الشفرات الرقم 85H الذي شفرته هي 10000101 ، ولزيادة المعلومات عن نظم العد والتشفير المتبع في الحاسب يمكن الرجوع إلى أي كتاب عن الإلكترونيات الرقمية . عندما يتعامل المعالج مع الذاكرة فإن وحدة التعامل بينهما تتوقف على عدد خطوط مسار البيانات لأن كل بت من باتات المعلومة تنقل على خط منفصل . في حالة الشرائح 8 بت فإن أي معلومة تنقل من أو إلى المعالج لابد وأن تكون مكونة من ثمانية باتات ، إذا كانت هذه المعلومة مكونة من عدد من البتات أكبر من ثمانية فإنها تنقل على أكثر من مرة وعلى حسب عدد باتاتها . في حالة الشرائح 16 بتا تكون وحدة التعامل في نقل المعلومات هي 16 بتا ، لذلك فإنه من البديهي أن تتوقف أن الشرائح 16 بتا تكون أسرع من الشرائح 8 بت لهذا السبب أساسا وأسباب أخرى سمعنا عنها فيما بعد ، فما بالك الآن بالشرائح 32 بتا والشرائح 64 بتا . لاحظ أن زيادة عدد باتات مسار البيانات لن ينعكس فقط على سرعة التعامل مع الذاكرة ولكنه ينعكس أيضا على سرعة تنفيذ العمليات الحاسوبية . كلمة أخيرة عن مسار البيانات وهي أن الإشارة عليه يمكن أن تكون خارجة من المعالج إلى الأجهزة المحيطة أو داخلة إلى المعالج من الأجهزة المحيطة .

2-5-3 خطوط التحكم Control lines

هذه الخطوط يختلف عددها من معالج لآخر وعن طريق هذه الخطوط يخبر المعالج أي جهاز من الأجهزة المحيطة (الذاكرة مثلا) الذي تم تحديد عنوانه على مسار العنوانين عن الغرض من هذا التعامل ، فقد يكون الغرض من التعامل مع الذاكرة مثلا هو القراءة منها ، أي استقبال معلومة منها ، في هذه الحالة فإن البروسيسور يرسل إشارة إلى الذاكرة على خط التحكم Memory Read، MEMR تعرف منها الذاكرة أن الغرض من التعامل هو القراءة فتقوم بإرسال المعلومة المطلوبة على مسار البيانات فينلقاها المعالج . أما إذا كان الغرض من التعامل هو الكتابة أو إرسال معلومة إلى الذاكرة فإن المعالج يقوم بوضع إشارة على الخط Memory Write, MEMW تفهم منها الذاكرة الغرض من التعامل فتنطلق المعلومة من على مسار البيانات . هناك خطان للتحكم بنفس الطريقة للتعامل مع بوابات الإخراج والإدخال . هناك أيضا خطوط المقاطعة Interrupt التي بها تتم مقاطعة أي برنامج يجري تنفيذه وخطوط المسك HOLD التي بها يتم فصل البروسيسور عن المسارات لأغراض معينة .

كمية الذاكرة التي يمكن التعامل معها	عدد خطوط مسار العنوانين
2 بايت	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8
512	9
1024 واحد كيلوبايت (اكتب)	10
ك ب 2	11
ك ب 4	12
ك ب 8	13
ك ب 16	14
ك ب 32	15
ك ب 64	16
ك ب 128	17
ك ب 256	18
ك ب 512	19
ك ب (1 ميجابايت ، امب)	20
مب 2	21
مب 4	22
مب 8	23
مب 16	24
مب 32	25
مب 64	26
مب 128	27
مب 256	28
مب 512	29
مب (1 جيجابايت ، 1 جب)	30
جب 2	31
جب 4	32
جب 8	33
جب 16	34
جب 32	35
جب 64	36

جدول 2-3 عدد خطوط مسار العنوانين وكمية الذاكرة

هذه الأطراف وغيرها سيأتي الكلام بالتفصيل عنها فيما بعد نظرا لأن عددها وشكل الإشارة عليها يختلف من معالج لآخر . من أهم الأطراف التي يجب أن

نأخذ فكرة عنها هي طرف التزامن CLOCK وعلى هذا الطرف يتم إدخال نبضات كهربائية بمواصفات معينة وتردد معين يحدد على حسب نوع شريحة المعالج . هذه النبضات CLOCK هي ساعة التوقيت الخاصة بالمعالج حيث يحدد زمن تنفيذ أي عملية يقوم بها المعالج بعدد معين من هذه النبضات يجب ألا تتعادل ، ولذلك فإن تردد هذه النبضات يعتبر خاصية من الخواص التي يعرف بها المعالج حيث بها أساساً تحدد سرعة المعالج . في حالة المعالجات 8 بت يكون تردد التزامن CLOCK اثنين ونصف ميجاهرتز تقريباً قد تزيد أو تقل من معالج آخر . إن ذلك يعني أن زمن النبضة الواحدة حوالي نصف ميكروثانية تقريباً ، فإذا علمنا أن عملية جمع مسجلين مثلاً تتم بعد 7 من هذه النبضات فلين ذلك يعني أن عملية جمع المسجلين ستتم في زمن مقداره ثلاثة ونصف ميكروثانية ! ، مما بالك بالمعالجات التي تبلغ نبضات الساعة لها الآن 400 أو 500 ميجاهرتز . جدول 2-3 يبين عدد خطوط مسار العنويين ومسار البيانات في بعض المعالجات ، وكذلك سنة ظهور كل واحد منها . هذا الجدول يبيّن أيضاً كمية الذاكرة التي يمكن لكل معالج من هذه المعالجات أن يتعامل معها . يبيّن الجدول أيضاً تردد نبضات الساعة لكل معالج كمقياس لسرعة تنفيذ الأوامر . حاول دراسة هذا الجدول لتتبين التطور السريع في بناء المعالجات .

رقم المعالج	سنة الظهور	عرض المسجلات	مسار العنويين	مسار البيانات	المدى العنوانى	تردد نبضات الساعة
8080	1974	8 بت	16 بت	8 بت	64 ك بايت	2 م هرتز
8085	1976	8 بت	16 بت	8 بت	64 ك بايت	2 م هرتز
Z80	1977	8 بت	16 بت	8 بت	64 ك بايت	4-2 م هرتز
8086	1978	16 بت	20 بت	16 بت	1 م بايت	16-6 م هرتز
80186	1980	16 بت	20 بت	16 بت	1 م بايت	16-6 م هرتز
80286	1982	16 بت	24 بت	16 بت	16 م بايت	20-12 م هرتز
80386	1985	32 بت	24 بت	16 بت	16 م بايت	40-16 م هرتز
80486	1989	32 بت	32 بت	32 بت	4 ج بايت	66-25 م هرتز
Pentium	1993	32 بت	32 بت	64 بت	4 ج بايت	200-60 م هرتز
Pentium Pro	1995	32 بت	36 بت	64 بت	4 ج بايت	150-200 م هرتز

جدول 2-3 معلومات عامة عن المعالجات التي سيتناولها هذا الكتاب

6- شرائح المعالجات ذات 8 بت 8 bit microprocessors

سندق النظر في هذا الجزء على تركيب شريحتين من شرائح الجيل الثاني من المعالجات وهي الشريحة Intel8085 و Z80 ولقد اختيرت هذه الشريحة بالذات لأنها هي الأكثر استخداماً وكانت وما زالت الأسهل في التعلم والابساط في التركيب والأسهل لتقديم فكرة المعالج وكيفية عمله وبرمجة المتعلمين الجدد في هذا المجال .

1-6-1 الشريحة Intel8085

شكل (5-2) يبين المحتويات التفصيلية لشريحة المعالج 8085 ومن هذا الشكل يمكننا ملاحظة الآتي :

1. نلاحظ وجود الحد الأدنى من المسجلات والعدادات الذي ذكرناه من قبل وهو مسجل تراكم واحد A وعدد برمامج PC ومسجل ومشفر للأوامر IR ومسجل مكذبة SP ومسجل حالة SR بالإضافة لوحدة الحساب والمنطق وستة من المسجلات العامة .

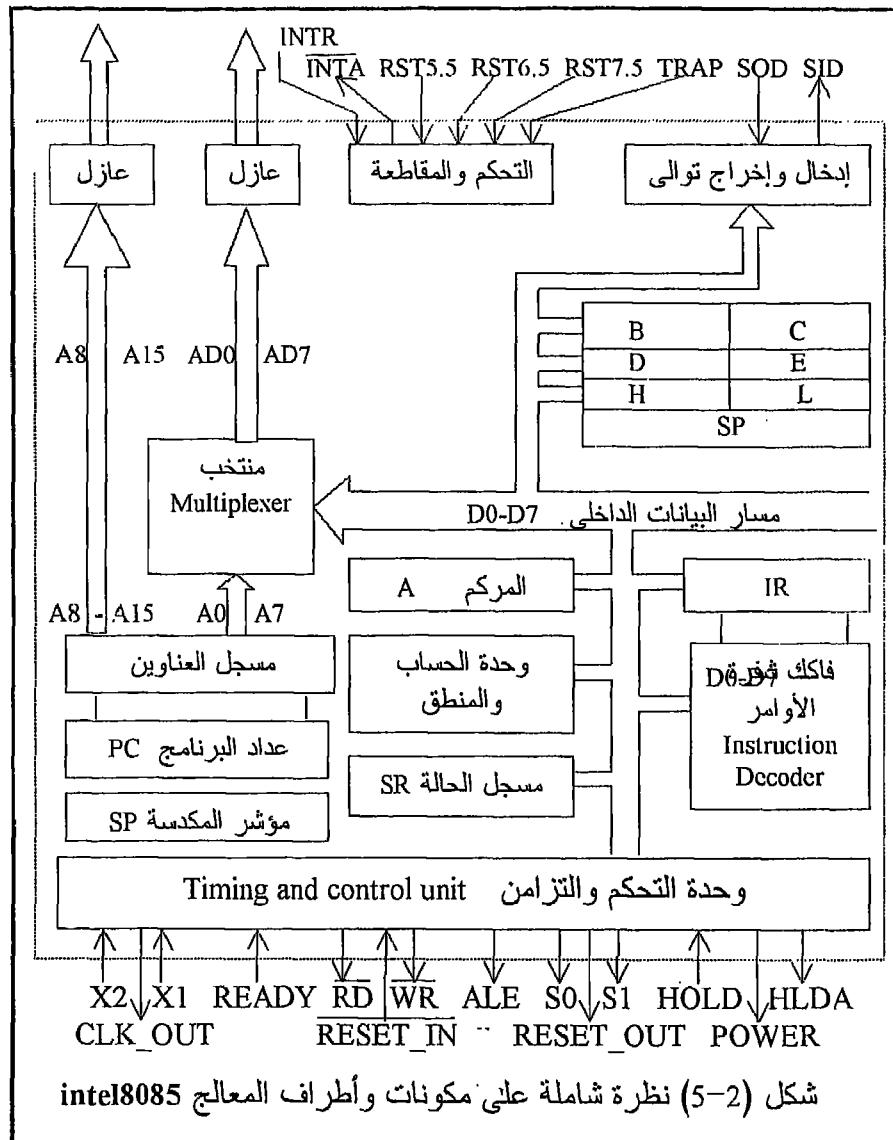
2. لاحظ وجود مسجل للعناوين Address register وهذا المسجل 16 بت يحتوى عنوان أى مكان في الذاكرة يراد التعامل معه . النصف العلوي من أى عنوان A15-A8 يخرج من مسجل العنوانين إلى خارج الشريحة مباشرة من خلال عازل Buffer ، وأما النصف الأول A7-A0 فإنه يدخل أولاً على مازج Multiplexer يقوم بدمج إشارة هذه الخطوط في تتبع زمني محدد مع الإشارة القادمة من مسار البيانات وإرسال الإشارتين على نفس الخطوط حيث تخرج إلى خارج الشريحة من خلال عازل أيضاً . إن عملية المزج هذه التي يقوم بها المازج يقصد بها تقليل عدد أرجل الشريحة وأما كيفية فصل الإشارتين ثانية فسوف يتم الحديث عنه بالتفصيل في فصل قادم .

3. عدد الأطراف الخارجية من الشريحة 40 طرفاً سيأتي الحديث عن كل طرف وشكل الإشارة الموجودة عليه في فصل قادم أيضاً إن شاء الله . الخط المنقط في شكل (5-2) عبارة عن حدود الشريحة يبين الأطراف الخارجية منها والتي من خلالها يتم الاتصال بين خارج الشريحة وداخلها . اتجاه السهم على هذه الخطوط يبين أيضاً اتجاه الإشارة على كل منها إذا كانت داخلة للمعالج أم خارجة منه .

2-6-2 المعالج Z80

كما نلاحظ من شكل (2-6) فإن أساس تركيب الشريحة Z80 هو نفسه أساس تركيب الشريحة Intel8085 من حيث وجود المسجلات الأساسية مثل عدد البرنامح ومسجل ومشفر الأوامر ومسجل التراكم وعدد من المسجلات العامة

وحدة الحساب والمنطق ووحدة التحكم والتزامن ، لكن كما ذكرنا من قبل دائمًا تكون هناك بعض الاختلافات عن هذا الأساس وتتمثل هذه الزيادات في حالة الشريحة Z80 فيما يلى :

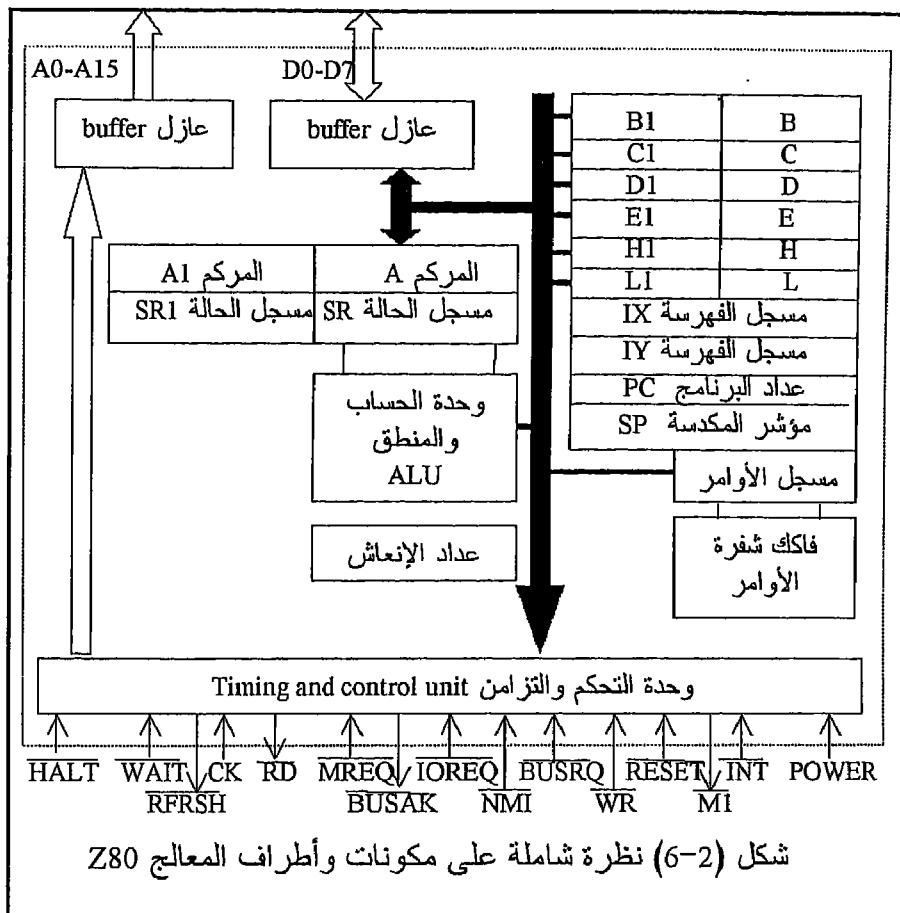


شكل (2-5) نظرة شاملة على مكونات وأطراف المعالج intel8085

- معظم مسجلات الشريحة Z80 تم مضاعفتها فهناك مثلاً مسجلين للستراكم A1 و A1 و مسجلين للحالة SR1 و جميع المسجلات العامة تم مضاعفتها

أيضاً كما هو موضح في شكل (2-6) وذلك سيكون له ميزة عظيمة في عملية البرمجة كما سنرى .

2. تم زيادة المسجلين IX و IY وكل منهما 16 بت وهذا المسجلان يستخدمان في طرق مختلفة لعنونة الذاكرة كما سيتضح فيما بعد .



شكل (2-6) نظرة شاملة على مكونات وأطراف المعالج Z80

3. تم زيادة عدد إنشاش الذاكرة والذى يستخدم فى عملية إنشاش الذاكرة الديناميكية لأنه كما نعلم فإن الذاكرة الديناميكية تحتاج دائماً لعملية تجديد أو إنشاش أو إعادة تخزين لمحطوياتها بعد فترات زمنية محددة وإلا فإنها تفقدتها بعد زمن مقداره بعض ميللائية .

4. نظرة شاملة على الأطراف الخارجية للمعالج Z80 سنجد أن له عدد 40 طرفاً لها تقريبا نفس الوظائف الخاصة بأطراف المعالج Intel8085 وإن اختلفت المسميات وسنعرف ذلك بالتفصيل عند دراسة الخواص والوظائف المختلفة لكل

طرف . الخط المنقط في شكل (2-6) عبارة عن حدود للشريحة بين الأطراف الخارجية منها والتي من خلالها يتم الاتصال بين خارج الشريحة وداخلها . اتجاه السهم على هذه الخطوط يبين اتجاه الإشارة على كل منها إذا كانت داخلة للمعالج أم خارجة منه .

7-2 تمارين

1. اذكر المهام الأساسية التي من المفروض أن يقوم بها أي معالج ؟
2. لماذا يعتبر مسجل التراكم من أهم المسجلات في المعالج ؟
3. كم عدد برات مسجل التراكم في المعالجات التي ندرسها في هذا الكتاب ؟ ولماذا هذا العدد بالذات ؟ وماذا يحدث لو نقص هذا العدد أو زاد ؟
4. هل تصنف مسجل التراكم من المسجلات عامة الأغراض general purpose registers أم المسجلات خاصة الأغراض dedicated registers ؟
5. ما هي وظيفة عدد البرنامج PC ؟ وكم عدد براتاته ؟ ولماذا يرتبط عدد براتاته بعدد براتات مسار العنوانين ؟
6. ماذا يحدث لو أن عدد براتات عدد البرنامج كان ثانية بدلا من 16 في المعالجات التي ندرسها ؟ وماذا يحدث لو أن هذا العدد كان 20 مثلا ؟
7. ما هي وظيفة مسجل الأوامر IR ؟ وكم عدد براتاته ؟ وهل يرتبط هذا العدد بمسار البيانات data bus أم بمسار العنوانين address bus ؟
8. ما هي وظيفة مسجل الحالة SR ؟ اذكر الأعلام الموجودة في مسجل الحالة للمعالج الذي تهتم بدراسته في هذا الكتاب ، ومتى يكون كل علم من هذه الأعلام واحدا ومتى يكون صفراء ؟
9. هل مسجل الحالة ومسجل الأوامر وعدد البرنامج تصنف على أنها مسجلات عامة الأغراض أم خاصة الأغراض ؟
10. على ضوء المهام المنوطة بالمعالج ، هل تشعر أن أي من المسجلات السابقة يعتبر زائدا ويمكن الاستغناء عنه ؟
11. ما هي محتويات كل علم من الأعلام بعد إجراء العمليات التالية :

10101111	01101110	11011101
<u>11110001</u>	<u>AND</u>	<u>00101111-</u>

11110001	XOR	10011001+
<u>11110001</u>	<u>11110001</u>	<u>11110001</u>

12. هل يحتوى المعالج الذى تهتم بدراسته على مسجلات عامة الأغراض غير مسجل التراكم ؟ أذكر هذه المسجلات ، وما فائدتها ؟ وهل يمكن الاستغناء عنها ؟
13. وضح بالرسم تركيب المعالج الذى تهتم بدراسته ؟

14. الكلية التي ندرس بها فيها 200 عضواً هيئه تدريس ، مطلوب إعطاء شفرة ثنائية لكل واحد منهم ، وكم سيكون عدد بذات هذه الشفرة ؟ هذه الشفرة الثنائية ، هل يمكن التعارف على أنها بمثابة عنوان للشخص ؟
15. شفرة ثنائية مكونة من 5 بذات ، كم عدد العناوين التي يمكن تشفيرها بهذه العدد من البذات ؟
16. كم عدد الخطوط (البذات) في مسار العناوين في المعالج الذي تدرسه ؟
17. ما مقدار كمية الذاكرة التي يستطيع أن يتعامل معها هذا المعالج ؟
18. ما هو تأثير زيادة أو تقصان عدد الخطوط في مسار العناوين لأى معالج ؟
19. لدينا 16 راكباً نريد نقلهم من مكان إلى مكان آخر باستخدام أتوبيس يسع 8 ركاب فقط ، كم عدد المشاوير التي سيقوم بها الأتوبيس ؟ لو استخدمنا أتوبيس يسع 16 راكباً ويسير بنفس سرعة الأول ، كم سيكون عدد المشاوير ؟ وأى الوسائلتين أسرع ؟
20. لو شبهنا الأتوبيس بمسار البيانات للمعالج ، وسعة الأتوبيس بعدد البذات (الخطوط) في هذا المسار ، أيهما سيكون أفضل من حيث السرعة في نقل المعلومات ، المعالج ذو 8 بذات أم ذو 16 بذات ؟

الفصل الثالث

برمجة المعالج

Microprocessor Programming

1-3 مقدمة

بعد أن أخذنا فكرة عامة عن وظيفة ومهمة كل مسجل من مسجلات شريحة المعالج سنتعرض في هذا الفصل وبعض الفصول القادمة إلى كيفية برمجة هذه الشريحة وهذا هو الشق الأول من دراسة شرائح المعالجات كما ذكرنا ، وأما الشق الآخر وهو مواجهة المعالج فإن ذلك سيكون في فصول قادمة أخرى إن شاء الله . سنتعرض في هذا الفصل لدراسة الأفكار العامة عن لغة التجميع (الأسمبل) دون أن نخوض بالذكر أي شريحة معينة حيث سيعقب هذا الفصل فصل خاص بلغة الأسمبل والأوامر الخاصة بكل واحدة من شرائح المعالج Z80 و Intel8085 .

2-3 لغات الحاسب Computer languages

لغات الحاسوب يمكن تقسيمها إلى قسمين أساسين :

القسم الأول : وهو اللغات التي تعتمد على الماكينة machine dependent وكل لغة من هذا النوع تكون مصممة لماكينة معينة وما يتتوافق منها مع ماكينة معينة ليس بالضرورة أن يتوافق مع الماكينات الأخرى . من أمثلة هذا النوع من اللغات ، لغة الماكينة machine language ولغة الأسمبل assembly language حيث أن كل معالج له مجموعة الأوامر الخاصة به التي عادة لا تتوافق مع المعالجات الأخرى . فانت مثلاً إذا كتبت برنامجاً بلغة الأسمبل الخاصة بالشريحة MC6800 فإن هذا البرنامج لا يمكن أن ينفذ مع الشريحة Intel8085 أو الشريحة Z80 .

القسم الثاني : هو اللغات التي لا تعتمد على الماكينة machine independent و من أمثلة هذه اللغات جميع اللغات ذات المستوى العالي high level language مثل الفورتران والبسكلال و PL/C و PL/1 وغير ذلك من هذه اللغات التي تعد بالمئات الآن .

3-3 ما هو الأمر؟

الأمر معناه الكود أو الشفرة الثنائية التي تعطى للمعالج والتي على أثرها يقوم بعمل فعل معين . هذا الفعل قد يكون عملية جمع رقمين أو إحضار معلومة من الذاكرة أو غير ذلك من الأفعال التي يستطيع المعالج القيام بها . من أمثلة هذه الشفرات الثنائية الشفرة 10000000 والتي معناها اجمع محتويات المسجل B مع

مسجل التراكم A وضع النتيجة في المسجل A . كما نرى فإن المعالج يتعرف فقط على الشفرات الثنائية ولا يعرف أى نوع آخر من الشفرات سواء كانت حرفية أو ثنائية أو سلعية . لقد سبق تعريف كل من الأمر والبرنامج في الفصل الأول بصورة عامة ولكننا أعدنا تعريفهما في هذا الفصل بشيء من التفصيل .

4-3 ما هو البرنامج ؟

```

00111010
01100000
00000000
01000111
00111010
01100001
00000000
10000000
00110010
01100010
00000000

```

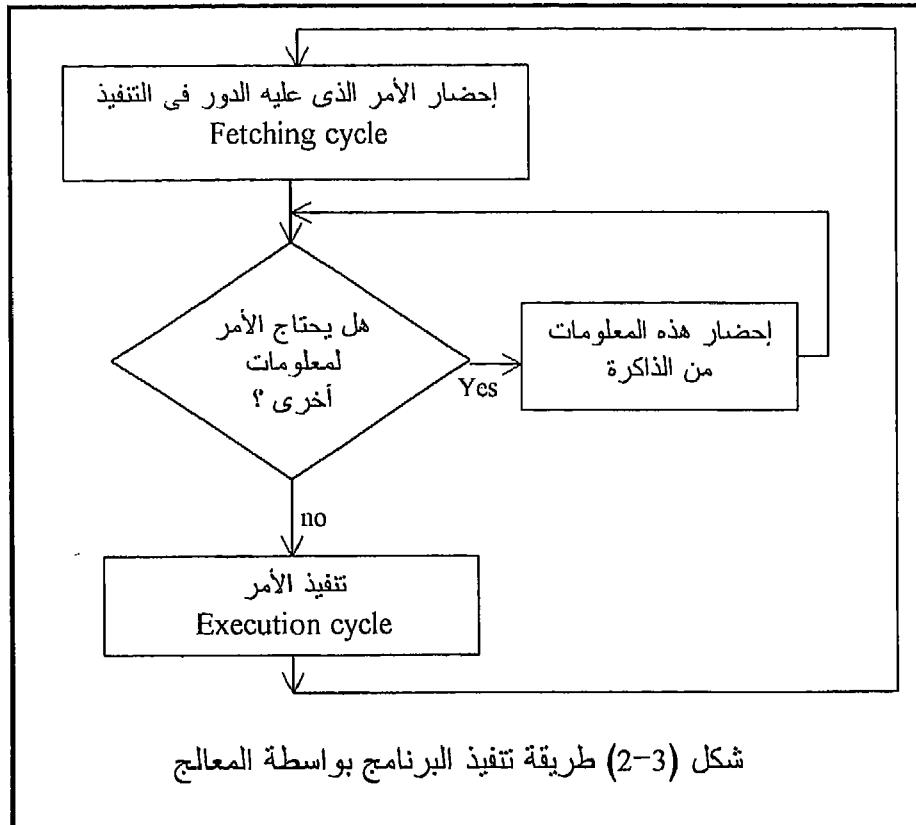
شكل (3-1) برنامج يجمع محتويات العنوان 60H مع العنوان 61H ويوضع النتيجة في العنوان H 62H

البرنامج عبارة عن مجموعة من الأوامر التي ينبع عن تفاصيلها مجتمعة هدف أو عمل معين كإدارة موتور مثلاً أو التحكم في متغير معين أو التعرف على معلومة معينة من بين الكثير من المعلومات . يجب أن يحتوى البرنامج أو يحدد مصدر أي معلومة يتم استخدامها بواسطته ، فمثلاً لإجراء عملية جمع لرقمين لأبد وأن يحدد البرنامج أين يوجد الرقمان وأين ستوضع النتيجة . يمكن النظر لأى برنامج على أنه مجموعة من الشفرات الثنائية المخزنة في الذاكرة في انتظار أن يقوم المعالج بتنفيذها . كمثال على ذلك انظر إلى البرنامج الموجود في شكل (3-1) دون أن تبذل أى مجهود في محاولة فهمه الآن . إن هذا البرنامج يقوم بجمع محتويات عنوان الذاكرة رقم 60H مع محتويات العنوان رقم 61H ويوضع النتيجة في العنوان H 62H ، إن الحرف H يعني أن هذه الأرقام مكتوبة بالنظام السبعدي . هذا البرنامج نقول أنه مكتوب بلغة الماكينة machine language وعادة يطلق عليه اسم برنامج الهدف object program .

3-5 كيف يقوم المعالج بتنفيذ البرنامج ؟

شكل (3-2) يبين الخطوات التي يقوم بها المعالج لكي يتم تنفيذ أي برنامج وهي كالتالي :

1. يقوم المعالج (وحدة التحكم بداخله) بقراءة الأمر الأول من الذاكرة وتخزينه في مسجل الأوامر IR .



2. يقوم المعالج بفك شفرة هذا الأمر أو بمعنى آخر يتم التعرف على هذا الأمر من بين قائمة أوامر المعالج / وعلى ضوء هذا التعارف يقرر المعالج إذا كان هذا الأمر سيحتاج لمعلومات أخرى من الذاكرة لكي تتم عملية التنفيذ أم لا ؟ وإذا كان الأمر سيحتاج لمثل هذه المعلومات يقوم المعالج بإحضارها أيضاً من الذاكرة. بذلك تنتهي المرحلة الأولى من مراحل تنفيذ الأمر الأول وهي مرحلة الإحضار fetching cycle .

3. بمجرد الانتهاء من مرحلة أو دورة الإحضار تبدأ مرحلة التنفيذ execution cycle حيث يقوم مشفر الأوامر مع وحدة التحكم بإرسال الإشارات المناسبة إلى وحدة الحساب والمنطق التي تقوم بتنفيذ هذا الأمر .

4. بعد الانتهاء من مرحلة تنفيذ الأمر الأول يرجع المعالج إلى الخطوة الأولى حيث يبدأ في عملية إحضار الأمر الثاني ثم يتم تنفيذه ثم يبدأ في عملية إحضار الأمر الثالث وتنتهي وهكذا حتى ينتهي البرنامج .

6-3 طريقة كتابة البرنامج للمعالج

3-1 الشفرات الثنائية Binary codes

إن الناظر لأول وهلة في البرنامج المكتوب بلغة الماكينة في شكل (3-1) سيصاب بالذهول وسيقول : هل من المعقول أن أكتب كل هذا العدد من الوحدات والأصفار في برنامج لا يتعدى الأربع خطوات ؟ بالطبع إن ذلك حق لأن كتابة البرامج بلغة الماكينة تصاحبها بعض العيوب والتي نوردها فيما يلى :

1. هذه البرامج تأخذ وقتا طويلا في إدخالها للذاكرة لأننا نكتبها بت بعده بت .
2. مثل هذه البرامج من الصعب فهمها أو متابعتها أو تصحيح أي خطأ فيها وذلك لأن الأعداد الثنائية عبارة عن نماذج من الوحدات والأصفار التي يصعب التفريق بينها خاصة بعد فترة عمل طويلة مع هذه الأرقام .
3. شكل هذا البرنامج لا يعطي أي دلالة على الغرض منه ، على العكس من برامج الباسيك أو حتى الأسمبلي كما سنرى بعد قليل فإنه بعد نظرة فاحصة على البرنامج تستطيع أن تخبر ما الغرض منه .
4. من السهل أن يقع المبرمج في الكثير من الأخطاء أثناء كتابة هذه البرامج ومن الصعب عليه جدا استخراج هذه الأخطاء فيما بعد .

كمثال على ذلك سنكتب البرنامج السابق الموجود في شكل (3-1) مرة أخرى في شكل (3-3) وبجانبه صورة منه تحتوى على خطأ معين وحاول استخراج هذا الخطأ !! ما رأيك الآن لو كان البرنامج مكونا من عشرات أو حتى مئات من السطور هل تستطيع استخراج أخطائه كلها ؟ لاحظ أنك في شكل (3-3) أمامك الصورة الصحيحة والصورة الخطأ وانت فقط تقارن الاثنين ولكن عادة في الوضع الحقيقي فإنه لن تكون أمامك الصورة الصحيحة للبرنامج ولكنك أخبرت بأن البرنامج به خطأ وعليك استخراجه ، بالطبع فإن هذه ستكون عملية شاقة .

3-2 الشفرات الستعشرية Hexadecimal codes

من الممكن تسهيل عملية كتابة البرامج بلغة الماكينة عن طريق استخدام نظام آخر غير النظام الثنائي ول يكن مثلا النظام الستعشرى أو النظام الثمانى وسنعرض

هذا للنظام الستعشري فقط على أساس أنه الأكثر شيوعا وأنه الأسهل في عملية الكتابة لأن عدد خانات العدد بالنظام الستعشري تكون عادة أقل منها في النظام الثنائي .

00111010	00111010
01100000	01100000
00000000	00000000
01000111	01000111
01110010	00111010
01100001	01100001
00000000	00000000
10000000	10000000
00110010	00110010
00110010	00110010
00000000	00000000

شكل (3.3) صعوبة استخراج الأخطاء في ظل الكتابة بلغة الماكينة

شكل (4-3) يبين برنامج الجمع السابق وقد تمت كتابته هذه المرة بالنظام الستعشري . من هذا الشكل نلاحظ أن عملية كتابة البرامج باستخدام النظام الستعشري وكذلك عملية فحص البرنامج واستخراج الأخطاء منه ستكون أسهل بكثير من استخدام النظام الثنائي في ذلك . المشكلة الآن هي أنه كما سبق وذكرنا أن المعالج لا يعرف سوى الإشارات المكتوبة بالنظام الثنائي فقط (وحابد وأصفار) فما هو الحل وقد كتبنا البرنامج بالنظام الستعشري كما هو في شكل (4-3) ؟ إن الحل لهذه المشكلة هو تحويل هذه الأوامر من الصورة الستعشورية إلى الصورة الثنائية قبل أن يتم إدخالها في الذاكرة !! ولكن من سيقوم بعملية التحويل هذه ؟ بالطبع إذا قام بها المستخدم فقد رجعنا إلى المشكلة الأولى وذلك لصعوبة عملية التحويل . إن هذه المهمة ، مهمة التحويل من الصورة الستعشورية إلى الصورة الثنائية ، هي مهمة سهلة جدا لأن يقوم بها المعالج نفسه عن طريق كتابة برنامج بلغة الماكينة (في النظام الثنائي) يتلقى الأوامر من المستخدم بالنظام الستعشري ثم يقوم هو (البرنامج) بتحويلها إلى النظام الثنائي وتحميلها في الذاكرة . إن هذا البرنامج يسمى "محمل النظام الستعشري" hexadecimal loader .

لقد حل النظام الستعشري مشكلة صعوبة كتابة البرامج واستخراج الأخطاء منها إلى حد ما ، ولكن بقيت المشكلة الأخرى وهي أننا ما زلنا نتعامل مع أرقام صماء

كشفرات للأوامر لا تحمل أي دلالة عن ماذا يفعل هذا الأمر أو ذاك . فمثلا الرقم 3A هو شفرة لأمر معين ولكننا لا نستطيع مثلا أن نميز ذلك الأمر فقد يكون هذا الرقم مثلا جزءا من عنوان كالأرقام 60, 61, 62 وغيرها ، وحتى إذا عرفنا أنه شفرة لأمر فلن نستطيع معرفة ماذا يفعل هذا الأمر إلا إذا رجعنا إلى كتالوج خاص بذلك .

3A
60
00
47
3A
61
00
80
32
62
00

شكل (3-4) نفس البرنامج الموجود في شكل (3-3) ولكن مكتوب
بالنظام السعدي

3-6 الشفرات الحرفية Mnemonics codes

إنها لفكرة عظيمة لو أتنا فهمنا المقصود من كل أمر من الأوامر وأعطيينا كل واحدا منها كودا أو شفرة مكونة من ثلاثة أو أربعة أحرف على الأكثر على أن تكون هذه الأحرف من الأحرف الأبجدية التي تدل تقريبا على ما يقوم به المعالج عند تنفيذ هذا الأمر . فمثلا أمر الجمع يكون ADD التي هي اختصارا ل الكلمة Addition يعني جمع ، وأمر الطرح يكون SUB و جاءت من Subtraction بمعنى طرح وهكذا مع باقي الأوامر كما سنرى فيما بعد . إن هذه الاختصارات هي ما يسمى بلغة الأسsembli Assembly language أو أحيانا تسمى Mnemonics codes بمعنى الشفرات التي من السهل تذكرها حيث كلمة mnemonics تعنى المساعد لعملية التذكر ، وهي كذلك فى الحقيقة ، إذ الآن بوضع الأوامر فى هذه الصورة الحرفية أصبح من السهل تذكرها بل ومن السهل أن تخبر ماذا يفعل الأمر بمجرد النظر إليه . ما أروعها لو أن العرب قد سبقوا فى هذا المجال وفرضوا الكلمات طرح وجمع بدلا من ADD و SUB !.....

يقوم كل صانع لشريحة من شرائح المعالج بتزويدها بقائمة أو كتالوج يحتوى كل هذه الاختصارات الحرفية mnemonics ، ولذلك فإنك ستجد أن اختصارات كل

شركة منتجة تختلف عن اختصارات الشركات الأخرى وسوف نرى ذلك في الفصول القادمة إن شاء الله . إن أي برنامج مكتوب بهذه الاختصارات يقال عنه أنه مكتوب بلغة الأسمبل . شكل (3-5) يبين البرنامج الموجود في شكل (3-4) والذي سبق كتابته بالشفرات الستعشرية وقد كتبت جميع أوامر هذه المرة بلغة الأسمبل لأحد المعالجات ، انظر لها البرنامج وحاول توقع ماذا يفعل كل أمر من هذه الأوامر قبل أن ندرسها بالتفصيل .

نحن هنا أيضاً أمام مشكلة ترجمة هذه الشفرات الحرفية mnemonics التي لا يستطيع المعالج التعرف عليها إلى شفرات ثنائية يعرفها المعالج ، وكما هي الحال مع الشفرات الستعشرية فإننا سنترك أمر هذه الترجمة ليقوم بها المعالج نفسه عن طريق برنامج مكتوب لهذا الغرض يقوم المعالج بتنفيذها فيحول هذه الشفرات الحرفية إلى الشفرات الثنائية المطلوبة . هذا البرنامج يطلق عليه الأسمبل Assembler . على ذلك نستطيع القول أن الأسمبل هو برنامج مكتوب بلغة الماكينة يقوم بتحويل البرنامج المكتوب بلغة الأسمبل (الشفرات الحرفية) إلى برنامج مكتوب بلغة الماكينة . عادة يطلق على البرنامج المكتوب بلغة الأسمبل "برنامج المصدر" source program والبرنامج المكتوب بلغة الماكينة "برنامج الهدف" object program . شكل (3-6) يبين رسمياً توضيحاً للدور الذي يقوم به الأسمبل .

إن المقابل الذي يدفعه المستخدم نتيجة استخدامه لغة الأسمبل يكون أولاً في كمية الذاكرة التي يشغلها برنامج الأسمبل حيث أن هذا البرنامج لابد وأن يشغل كمية من الذاكرة الأساسية للميكروكمبيوتر وثانياً بعض التأخير الذي يحدث نتيجة الوقت الذي يأخذه الأسمبل في عملية الترجمة . وكما نعرف فإن إيه ليس هناك شيء كامل على الإطلاق ، لذلك فإننا نستطيع أن نلخص بعض عيوب لغة الأسمبل فيما يلى :

1. مازالت هذه الاختصارات الحرفية غير كافية للدلالة على معنى الأوامر المختلفة حيث مازالت صيغ الأوامر بعيدة كل البعد عن اللغة العادية التي يستخدمها الإنسان وأيضاً بالمقارنة بأوامر اللغات ذات المستوى العالي فإن لغة الأسمبل تعتبر الأصعب في التعلم .
2. لكي تستخدم هذه اللغة لابد من المعرفة الكاملة بمكونات المعالج ، المسجلات الموجودة بداخلة ، وطريقة المعالج في التعامل مع الذاكرة وغير ذلك من الأمور الغير موجودة في اللغات ذات المستوى العالي .
3. هذه اللغة كما ذكرنا من قبل تعتبر من اللغات التي تعتمد على الماكينة ، فأنتم إذا كتبت برامجاً للمعالج Z80 فلن تستطيع استخدامه مع المعالج MC6800 مثلاً.

```

LDA
60
00
MOV B,A
LDA
61
00
ADD B
STA
62
00

```

شكل (3-5) البرنامج الموجود في شكل (3-4) وقد كتب هذه المرة بلغة الأسمبل

برنامـج مكتوب
بلغـة الأسمـبلـى
برنـامـج المصـدرـ
Source program

"برنـامـج الأـسمـبلـرـ"
Assembler
program
الأـسمـبلـرـ

برنـامـج مكتوب
بلغـة المـاكـيـنةـ
برنـامـج الـهـدـفـ
Object program

شكل (3.6) مهمة الأسمبلر

7-3 اللغات ذات المستوى العالى High level languages

لقد تم التغلب على الكثير من الصعوبات والعيوب المصاحبة لغة الأسمبل باستخدام اللغات ذات المستوى العالى . إن كل أمر من أوامر أي لغة من هذه اللغات يدل أو يقوم بعملية مركبة على عكس لغة الأسمبل فإن كل أمر فيها يقوم بعملية أولية . فمثلا برنامج الجمع السابق الذى يجمع رقمين موجودين فى الذاكرة والذى تمت كتابته فى أحد عشر سطرا باستخدام لغة الأسمبل يمكن كتابته فى سطر واحد باستخدام لغة الباسيك كما يلى :

SUM = NUM1 + NUM2

لذلك فإن أي أمر من أوامر أي لغة من اللغات ذات المستوى العالى هو فى الحقيقة مجموعة من أوامر لغة الأسمبل . إن ما يقوم به برنامج الأسمبل فى حالة لغة الأسمبل يقوم به برنامج آخر يسمى "برنامج المؤلف" أو الجامع أو المصنف compiler program فى حالة اللغات ذات المستوى العالى ، حيث يقوم هذا المؤلف بترجمة الأوامر المكتوبة باللغات ذات المستوى العالى إلى لغة الماكينة أو الشفرات الثنائية التى يقبلها المعالج . هذه اللغات ليست موضوع دراستنا فى هذا الكتاب لذلك سنكتفى بهذا القدر من الكلام عنها .

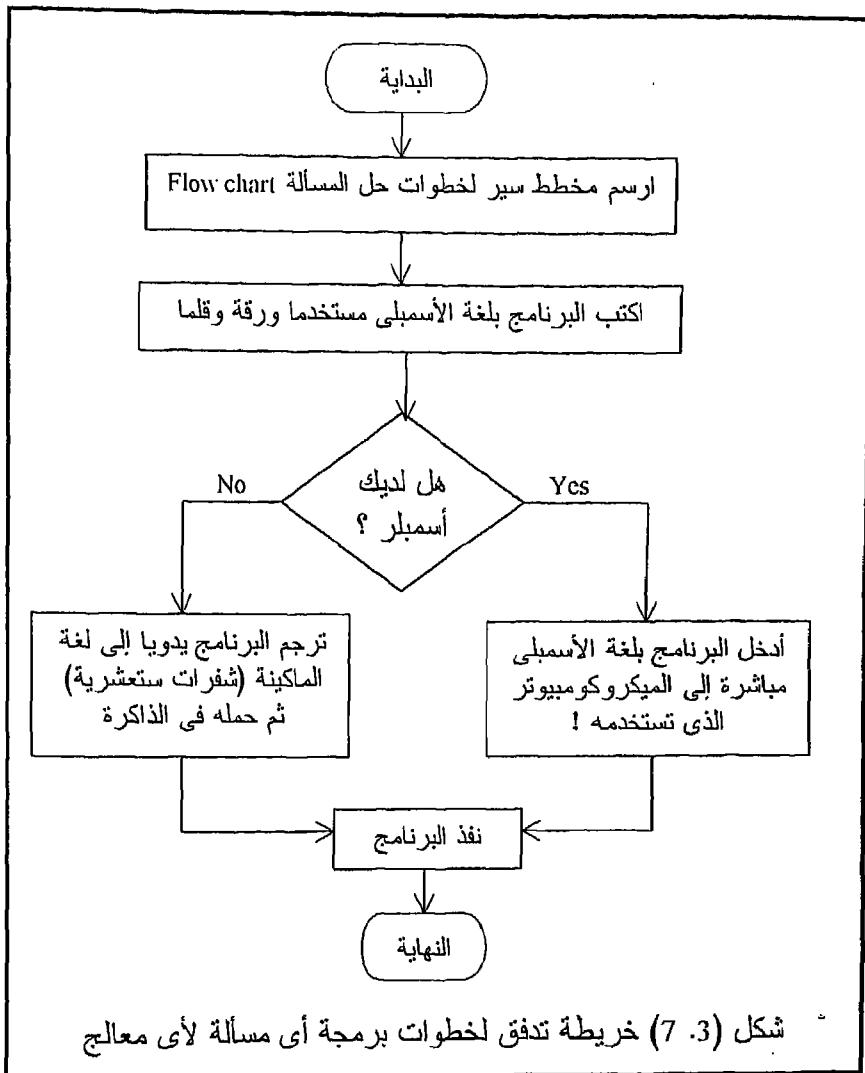
3-8 خطوات كتابة برنامج بلغة الأسمبل

الآن وقد عرفا الفرق بين لغة الماكينة ولغة الأسمبل فبأى صورة سنكتب برامجنا ؟ الإجابة على ذلك ستتوقف على إمكانيات الميكروكمبيوتر الذى تتعامل معه سواء فى بيتك أو فى معملك . إذا كان لديك برنامج الأسمبل فإنه بالطبع من الأفضل أن نكتب برامجك بلغة الأسمبل على الحاسب بنفس الشكل ثم طلب من الأسمبل أن يقوم بعملية الترجمة وإدخال البرنامج إلى الذاكرة ، أما إذا لم يكن لديك أسمبل فليس أمامك من خيار سوى الكتابة بلغة الماكينة أو الشفرات العشرية فى الذاكرة مباشرة . شكل (3-7) يبين خريطة تدفق أو مخطط سير flow chart لخطوات حل أي مشكلة باستخدام المعالج .

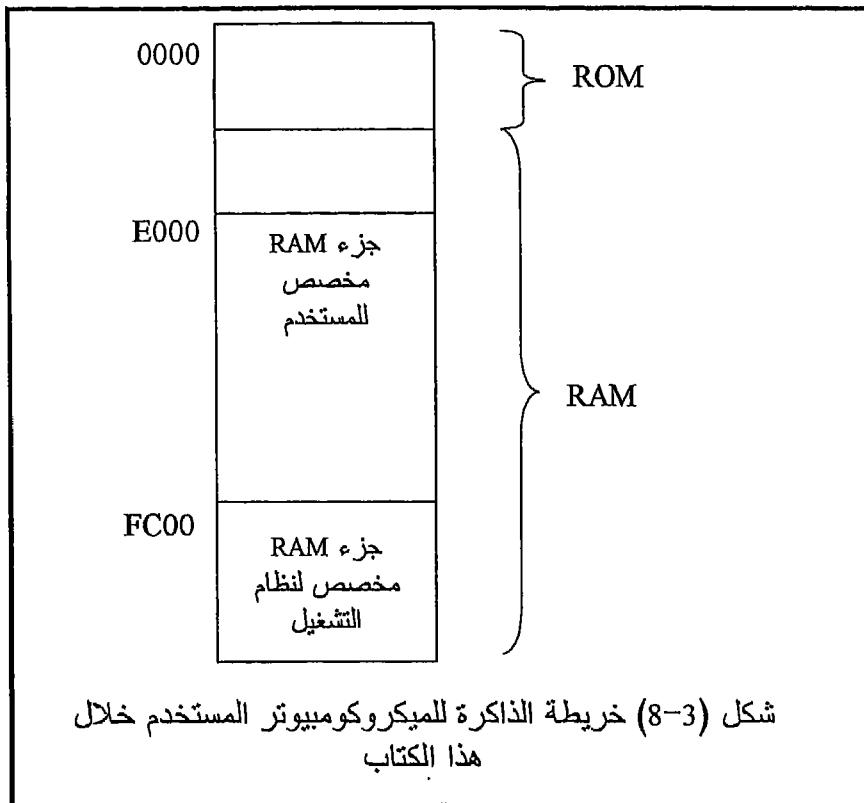
السؤال الآن فى أي مكان فى الذاكرة سنضع البرنامج ؟

للإجابة عن هذا السؤال يجب أن نتصور الذاكرة الخاصة بالجهاز أو الميكروكمبيوتر الذى نستخدمه وقد قسمت إلى ثلاثة أجزاء كالموضحة فى شكل (3-8) . الجزء الأول منها هو ROM أو ذاكرة القراءة فقط وهذه كما ذكرنا فى الفصل الأول لا يمكن للمستخدم أن يسجل فيها أى شيء . الجزء الثانى من الذاكرة هى RAM وهى الجزء الذى يمكن للمستخدم أن يتعامل معه ويجب أن نعلم أن نظام التشغيل الخاص بالجهاز الذى نستخدمه يحجز جزءا من هذه RAM للاستعمال الخاص به والجزء المتبقى يمكن للمبرمج أن يستخدمه . لذلك يجب قبل أن تبدأ فى كتابة برنامجك بلغة الأسمبل وإدخاله فى الذاكرة أن تعرف أين يقع جزء RAM الخاص بنظام التشغيل حتى لا يتداخل البرنامج الخاص بك معه . إن ذلك يتطلب إلقاء نظرة على ما يسمى بخريطة الذاكرة الخاصة بالميكروكمبيوتر الذى تستخدمه . هذه الخريطة تعتبر شكلاً توضيحاً يبين الذاكرة بأكملها من الأول حتى آخر بait وقد قسمت إلى أجزاء مع التعريف بكل جزء فيما يستخدم وهل هو مشغول أم لا . الآن وقد عرفت الجزء من RAM الذى يمكنك أن تضع فيه برامحك يجب عليك كمبرمج أن تقسّم هذا الجزء إلى

جزاين أيضا ، أحدهما تكتب فيه البرنامج والأخر تخصصه للبيانات التي يحتاجها أو يخرجها البرنامج .



إن عملية تقسيم الذاكرة إلى جزء للبرنامج وأخر للبيانات عملية تعتمد على المبرمج بالدرجة الأولى وعلى البرنامج أيضا ، فقد يكون البرنامج لا يحتاج إلى بيانات أو لا يخرج بيانات على الإطلاق ، في هذه الحالة فإن كل RAM ستكون للبرنامج ، وقد يكون البرنامج ينتج أو يحتاج للكثير من البيانات ، في هذه الحالة يجب حجز جزء كاف لهذه البيانات .



مثال 1.

لدينا مجموعة من الأرقام ولتكن عددها 50H رقما ، والمطلوب استخراج أكبر عدد في هذه المجموعة . أين سنكتب البرنامج ؟ وأين سنكتب البيانات (الخمسين رقم) ؟

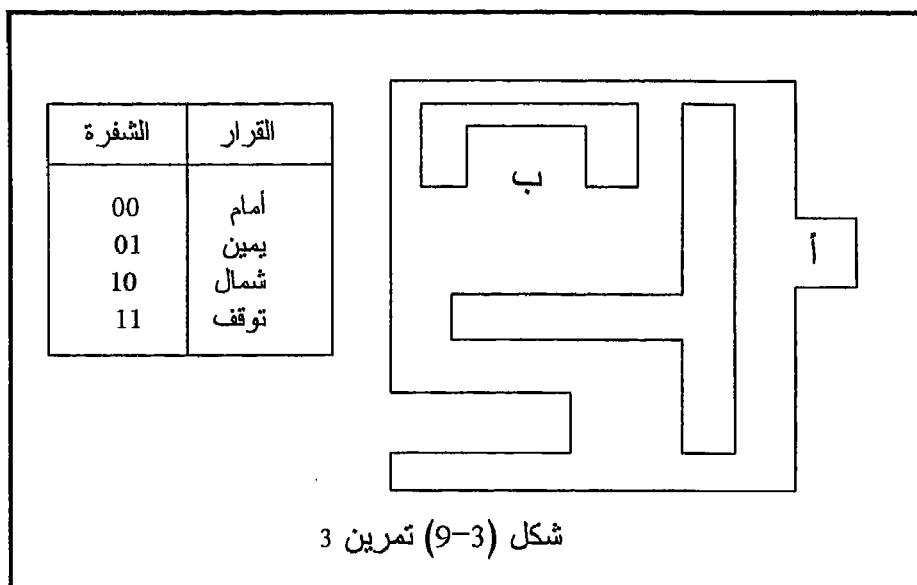
كما نرى من شكل (8-3) فإن جزء RAM المخصص للمستخدم في الميكروكمبيوتر الذي نستخدمه يبدأ من العنوان E000H وينتهي عند العنوان FC00H . هذه المساحة من RAM يجب أن نقسمها إلى جزء للبرنامج وجزء للبيانات . البيانات المطلوبة هي 50H رقما ويمكن لنا أن نضعها ابتداء من العنوان E100H حتى العنوان E150H . أما البرنامج فيمكن لنا أن نكتبه مثلا ابتداء من العنوان E000H على أساس أن البيانات التي سيحتاجها البرنامج نفسه لن يصل عددها إلى 100H بait بأى حال من الأحوال وإلا لو زاد عددها عن ذلك فسوف تتدخل مع البيانات عند العنوان E100H وفي هذه الحالة يجب أن نزيح البيانات بعيدا أو نكتب البرنامج بعد البيانات .

عملية تقسيم الذاكرة الموضحة في شكل (3-8) يسمى في مصطلحات الحاسوب بخريطة الذاكرة حيث يبين هذا الشكل جميع الذاكرة الملحقة بالجهاز الذي تستخدمه وفيما يستخدم كل جزء منها ، وهذه الخريطة يجب على أي مستخدم للغة الأsembler أن يتعرف عليها جيدا بالنسبة للحاسوب الذي سيستخدمه ، والتقسيم الموضح في شكل (3-8) سيكون هو التقسيم الذي ستبعه خلال هذا الكتاب وبالطبع فإن هذه الخريطة تختلف باختلاف الجهاز المستخدم فيجب مراعاة ذلك .

9- تمارين

1. ما هو الأمر ؟
2. ما هو البرنامج ؟
3. تخيل أنك تركب سيارة آلية تريد الانتقال بها من المكان (أ) إلى المكان (ب) كما في شكل (3.9) هذه السيارة عند كل تقاطع لابد أن تأخذ قرارا من أربعة تحدد اتجاه حركتها كما في الجدول الموضح في نفس الشكل . اكتب برنامجا لهذه السيارة بالقرارات ثم بالشفرات الثنائية بحيث عندما تتفوزه تنتقل السيارة من المكان (أ) إلى المكان (ب) في الشكل .
4. هل يمكنك إعطاء اسم لغة الشفرات الثنائية المكتوب بها البرنامج السابق ؟
5. ماذا يحدث لو جعلنا شفرا القرار تتكون من 3 بิตات بدلا من 2 ؟ بالطبع في هذه الحالة سيكون هناك إمكانية لعدد أكبر من القرارات يصل إلى 8 ، افترض هذه القرارات من عندك محاولا تطوير هذه السيارة ؟
6. اشرح باستخدام خريطة تتفق كيف يقوم المعالج بتنفيذ أي برنامج ؟
7. ما هي عيوب كتابة البرامج بالشفرات الثنائية ؟
8. أعد كتابة البرنامج الموجود في شكل (3-1) مستخدما النظام الثمانى ؟ كم عدد ضربات المفاتيح التي ستتفذها لكي تكتب البرنامج مستخدما هذا النظام ؟ ما نوع المحمول loader الذى ستحتاج إليه فى هذه الحالة ؟ أيهما أفضل فى الكتابة وسهولة استخراج الأخطاء ، النظام الثمانى أم النظام الستعشري ؟
9. اشرح فائدة استخدام الأsembler ؟ ...
10. ما هي عيوب البرمجة باستخدام لغة التجميع مقارنة باللغات ذات المستوى العالى ؟
11. اشرح الفرق بين الأsembler والمترجم ؟
12. هل لغة C من اللغات التى تعتمد على الماكينة ؟
13. مطلوب كتابة برنامج يجمع 20 رقمًا مخزنـة في الذاكرة ، ارسم خريطة للذاكرة تبين عليها أين ستكتب البرنامج وأين ستوضع البيانات ؟

14. حاول الحصول على خريطة الذاكرة للميكروكمبيوتر الذي تتعامل معه
وادرسها وطبق عليها المسألة السابقة؟



الفصل الرابع

برمجة المعالج Intel 8085

*Programming The Intel 8085
Microprocessor*

١-٤ مقدمة

لبرمجة أي معالج لابد من دراسة مجموعة الأوامر الخاصة به ولكن نسهل دراسة هذه الأوامر سنقوم بتنسيقها إلى مجموعات من حيث الوظيفة التي يؤديها كل أمر وسندرس بالتفصيل في كل مجموعة بعض الأوامر الكثيرة الاستخدام مع التمثيل ببعض الأمثلة ، على أننا سنعرض في نهاية الفصل لجدول تحتوى على جميع أوامر الشريحة موضوعة في مجموعات ثم سنعرض أيضا جدول يحتوى على هذه الأوامر مرتبة أبجديا مع نبذة بسيطة عن وظيفتها كل أمر وشفرته .

٢-٤ مجموعة أوامر الانتقال Transfer instructions

يقوم أي أمر من أوامر هذه المجموعة بنقل معلومة من مكان آخر حيث المكان الذي تخرج منه المعلومة يسمى بالمصدر Source وسنرمز له بالرمز sss وهذا المكان قد يكون مسجلا داخل شريحة المعالج وقد يكون مكانا من أماكن الذاكرة . وأما المكان الذي ستذهب إليه المعلومة فسوف نسميه الهدف Destination وسنرمز له بالرمز ddd وهذا المكان أيضا قد يكون مسجلا داخل شريحة المعالج وقد يكون بآيت من بآيات الذاكرة كما سنرى . شكل (٤-١) يبين أهم الأوامر الموجودة في مجموعة أوامر الانتقال الخاصة بالشريحة 8085 والتي تهمنا في هذه المرحلة من دراسة لغة الأسsembl .

MOV
MVI
LXI
LDA
STA
LHLD
SHLD

شكل (٤-١) بعض أوامر الانتقال الكثيرة الاستخدام للشريحة 8085

١-٢-٤ الأمر MOV

الصورة العامة لهذا الأمر هي :

MOV ddd,sss

ddd ← مسجل sss

ومعنى هذا الأمر انقل أو حرك (وهذا هو ترجمة كلمة move) المعلومة الموجودة في المصدر sss إلى الهدف ddd ، لاحظ أن المعالج عندما يقوم بنقل معلومة فإنه ينقل صورة منها فقط أما أصل المعلومة فيظل في المصدر ولا يتغير . الصورة العامة لشفرة هذا الأمر الثانية يمكن كتابتها كما يلى :

01dddsss

حيث sss تستبدل بشفرة مصدر المعلومة سواء كانت مسجلأ أم ذاكرة و ddd تستبدل أيضا بشفرة الهدف الذي ستلتجأ إليه المعلومة سواء كان مسجلأ أم ذاكرة . لاحظ أن هذا الأمر يتكون دائما من بaitt واحدة . راجع شفرات المسجلات فى الفصل الثاني ، جدول 2-1 ، وانظر المثال التالى :

مثال 1-4

1. الأمر MOV A,B

شفرته الثنائية هي 01111000

شفرته الستعشرية هي 78H

هذا الأمر سينقل محتويات المسجل B (صورة منها فقط) إلى المسجل A لاحظ أن المسجل B هو المصدر sss ولذلك استبدلناه بشفرته الثنائية وهي 000 والمسجل A هو الهدف ddd واستبدلناه بشفرته الثنائية 111 لتصبح شفرة الأمر الثانية هي 01111000 أو H 78 فى النظام الستعشرى ، ولمزيد من الأمثلة إليك ما يلى :

2. الأمر MOV L,D

شفرته الثنائية هي 01101010

شفرته الستعشرية هي 6AH

3. الأمر MOV C,C

شفرته الثنائية هي 01001001

شفرته الستعشرية هي 49H

هذا الأمر ينقل محتويات المسجل C إلى نفسه وهذا يكفى تماما ، لا تعمل شيئا . مثل هذه الأوامر التي لا تعمل شيئا لها أهمية كبيرة في الكثير من التطبيقات كما سيأتي فيما بعد .

في جميع الأوامر السابقة كنا ننقل المعلومة من مسجل إلى مسجل آخر ، ماذا لو أردنا نقل معلومة من مسجل إلى الذاكرة أو العكس . المثال التالي سيوضح ذلك .

مثال 2-4

1. الأمر MOV M,A

شفرته الثنائية هي 01110111

شفرته السبعية هي 77H

هذا الأمر ينقل محتويات المسجل A وهو مصدر المعلومة إلى الذاكرة M وهي الهدف الذي ستدهب إليه المعلومة ، لاحظ أن M استبدلت بالشفرة 110 كما في جدول 2-1 . السؤال الآن هو : في أي مكان أو في أي عنوان في الذاكرة ستذهب محتويات المسجل A ؟ في جميع الأوامر التي تتعامل مع الذاكرة بهذه الشكل يكون العنوان موجودا في زوج المسجلات HL ، أي أن محتويات المسجل A ستذهب إلى بait الذاكرة التي يوجد عنوانها في المسجلين H و L . لاحظ أن هذه الطريقة هي ما سنسميه بطريقة التعامل غير المباشر مع الذاكرة عندما ستصنف طرق التعامل مع الذاكرة في نهاية هذا الفصل .

2. الأمر MOV B,M

01000110

46H

هذا الأمر سينقل محتويات بait الذاكرة التي يوجد عنوانها في المسجلين H و L إلى المسجل B .

MVI 2-2 الأمر

هذا الأمر معناه "انقل المعلومة الفورية" أي Move the Immediate data والصورة العامة له هي :

MVI ddd,data8

ddd ← data8

هذا الأمر يضع المعلومة المكونة من ثمانية بتات (data8) في الهدف . ddd الهدف ddd قد يكون مسجلا أو الذاكرة M كما سنرى في الأمثلة . هذا الأمر يتكون دائما من اثنتين من الباليتات . واحدة هي شفرة الأمر operation code واختصارا op code والباليت الأخرى هي المعلومة (data8) . وعلى ذلك ستكون الصورة العامة للشفرة الثنائية لهذا الأمر كالتالي :

00ddd110

data8

حيث ddd تستبدل بشفرة المسجل أو الذاكرة M المراد وضع المعلومة فيها .

مثال 3-4

MVI B,53H . 1

الشفرة الثنائية هي : 00000110

01010011

الشفرة الستعشرية هي : 06H

53H

هذا الأمر سيضع المعلومة الفورية أو الثابت 53H في المسجل B . نفضل أن نسمى مثل هذه المعلومة بالعلومة الفورية لأنها ليس لها مصدرا وإنما مصدرها هو المستخدم نفسه ، ومن هنا كان الحرف I في الأمر وهو اختصارا لكلمة Immediate أو فوري وسوف يصادفنا أوامر أخرى تحتوى الحرف I وكلها تتعامل مع معلومات فورية أو ثوابت بهذا الشكل .

من الممكن تحميل معلومة فورية Immediate في مكان ما في الذاكرة بحيث يكون عنوان هذا المكان في المسجلين HL كالتالى :

MVI M,data8 .2

والشفرة الثانية لهذا الأمر ستكون كالتالى :

00110110

data8

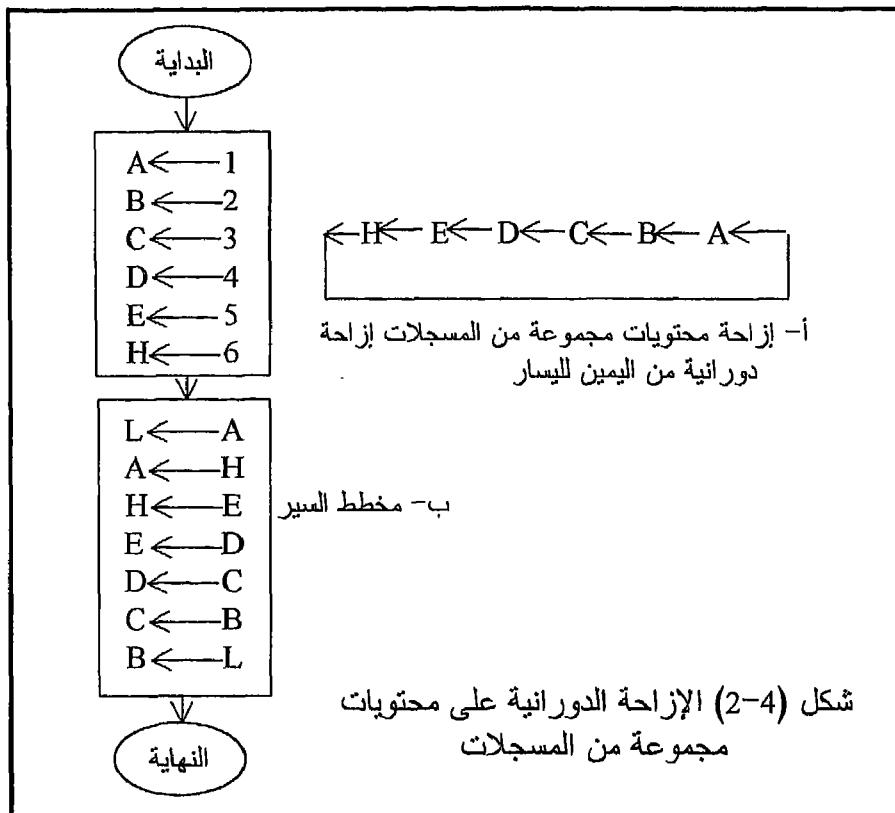
المثال التالي يعتبر تدريبا جيدا على الأمرين MOV و MVI

مثال 4-4

المطلوب تحميل المسجلات A, B, C, D, E, H بالمعلومات الفورية التالية : 01, 02, 03, 04, 05, 06 على التوالي ، ثم بعد ذلك يتم عمل إزاحة دورانية لهذه المحتويات كما هو مبين في شكل (4-2أ) بحيث أن محتويات المسجل A تذهب إلى المسجل B ومحتويات B تذهب إلى C وهكذا إلى أن تذهب محتويات المسجل H إلى المسجل A دون فقد محتويات أي مسجل . شكل (4-2ب) يوضح مخطط السير لهذا البرنامج ، وشكل (3-4) يبين الشفرات الأسمبلية والثانية والستعشرية للبرنامج . لتنفيذ هذا البرنامج يمكنك أن تفتح ال RAM عند العنوان E000 وتبدا في إدخال البرنامج إما بالشفرات الثنائية أو الشفرات الستعشرية ، وأما إذا كان الميكروكمبيوتر الذي تستخدمه به الأسمبلر الخاص بالشريحة 8085 فيمكنك الدخول فيه وكتابة البرنامج باستخدام شفرات الأسمبلية وستترك تفاصيل عملية إدخال البرنامج على الجهاز لأنها تختلف على حسب الإمكانيات ومن شخص لأخر . بعد الانتهاء من إدخال البرنامج يمكنك تنفيذه باستخدام الأمر GO E000 أو من الأفضل تنفيذه بطريقة الخطوة خطوة حيث يمكنك في هذه الحالة متابعة البيانات أثناء انتقالها من مسجل لأخر .

عند استخدام الأمر MOV M,C مثلا لنقل محتويات المسجل C إلى الذاكرة أو الأمر MVI M,33 لتحميل المعلومة الفورية 33 في الذاكرة ذكرنا أن عنوان الذاكرة الذي سيتم التعامل معه يكون دائما في المسجلين HL . السؤال الآن كيف

نضع هذا العنوان في المسجلين H و L؟ الإجابة عن هذا السؤال توجد في الأمر
• LXI



شكل (4-2) الإزاحة الدورانية على محتويات مجموعة من المسجلات

3-2-4 الأمر LXI

الصورة العامة لهذا الأمر هي:

LXI rp,data16

rp——> زوج مسجلات

الشفرة الثانية للأمر كالتالي :

00rp0001

البايت ذات القيمة الصغرى من المعلومة data16

البايت ذات القيمة العظمى من المعلومة data16

حيث يقوم هذا الأمر بتحميل زوج المسجلات الذي توضع شفرته بدلاً من rp (في البايت الأولى للأمر) بالمعلومة الفورية data16 أي المكونة من 16 بت والتي توضع في الإثنين بايت التاليتين . حرف I الموجود في صورة الأمر له

نفس الدلالة التي عرفناها مسبقاً والتي تعنى فوري أى Immediate . لاحظ أن المعلومة مكونة من 16 بت وشفرة الأمر op code ستكون بait واحدة ، لذلك فإن هذا الأمر يتكون دائماً من ثلاثة بايتات كما سنرى في الأمثلة . لاحظ أيضاً أن هناك أربعة أزواج من المسجلات فقط يمكن استخدامها مع هذا الأمر وهي الإزواج SP , HL, DE, BC والتى سبق أن ذكرنا شفراتها فى جدول 1-2 . لذلك فإن البايت ذات القيمة العظمى من المعلومة الـ 16 بت تذهب دائماً إلى المسجل ذى القيمة العظمى من الزوج وهو المسجل B أو D أو H أو التصف الأعلى من المسجل SP والبايت ذات القيمة الصغرى تذهب إلى المسجل ذى القيمة الصغرى من الزوج وهو المسجل C أو E أو L أو التصف الأول من المسجل SP .

العناوين	شفرات أسمبلى	شفرات ثنائية	شفرات ستعشرية
E000	MVI A,01	00111110	3E
E001		00000001	01
E002	MVI B,02	00000110	06
E003		00000010	02
E004	MVI C,03	00001110	0E
E005		00000011	03
E006	MVI D,04	00010110	16
E007		00000100	04
E008	MVI E,05	00011110	1E
E009		00000101	05
E00A	MVI H,06	00100110	26
E00B		00000110	06
E00C	MOV L,A	01101111	6F
E00D	MOV A,H	01111100	7C
E00E	MOV H,E	01100011	63
E00F	MOV E,D	01011010	5A
E010	MOV D,C	01010001	51
E011	MOV C,B	01001000	48
E012	MOV B,L	01000101	45

شكل (3-4) برنامج الإزاحة الدورية

مثال 5-4

LXI H,2C3A	1. شفرة الأسمبلى
00100001	الشفرة الثانية
00111010	
00101100	
21	الشفرة العتشرية
3A	
2C	

يقوم هذا الأمر بتحميل زوج المسجلات HL بالمعلومة الفورية 2C3A بحيث ستدهب البايت ذات القيمة العظمى وهي 2C إلى المسجل H والبايت ذات القيمة الصغرى وهي 3A إلى المسجل L . بنفس الطريقة يمكن تحميل الأزواج الأخرى من المسجلات . كما رأينا فإن هذا الأمر مهم جدا عند التعامل مع الذاكرة حيث عنوان المكان المراد التعامل معه يوضع في المسجلين H و L باستخدام هذا الأمر .

مثال 6-4

حمل مكان الذاكرة E100 بالمعلومة 66H . أحد الطرق لكي يتم ذلك هي أن نقوم بتحميل العنوان E100 في زوج المسجلات HL ثم نستخدم الأمر MVI كما يلى :

E000, E001, E002 LXI H,E100
E003, E004 MVI M,66H

لاحظ أننا استخدمنا هنا الشفرات الأسمبلى فقط ولاحظ أيضا أن الأمر LXI يشغل ثلاثة بايتات كما ذكرنا وهي البايتات E000, E001, E002 .

إن الطريقة السابقة في التعامل مع الذاكرة والتي تستخدم زوج المسجلات HL كوسiel يحمل العنوان المراد التعامل معه تعتبر بل وتسى بالطريقة غير المباشرة في التعامل مع الذاكرة حيث أن المعالج قبل أن يذهب إلى الذاكرة سواء للقراءة أو للكتابة لابد وأن يمر أولا على زوج المسجلات HL ليعرف منهما العنوان الذى سيذهب إليه . إن ذلك تماما مثلما أنك تقول لصديقك محمد يا أخي يا محمد وأنت مسافر إلى الرياض خذ هذه الرسالة وأعطيها لأخي أحمد ولكنى لا أعرف عنوانه فرجاء أن تذهب إلى أخي الأكبر محمود وهو يسكن معنا هنا فتعرف منه عنوان أحمد قبل أن تسفر إلى الرياض . صديقك محمد فى هذا المثال يمثل المعالج الذى سيقوم بالتنفيذ وأما أخيك الأكبر محمود فيمثل المسجلين HL الذين عندهما عنوان أخيك أحمد الذى يمثل بait الذاكرة المراد التعامل معها . نعيد التأكيد هنا أن زوج المسجلات HL فقط هما اللذان يحتويان عنوان المكان المراد التعامل معه في مثل هذه الأوامر .

هناك طريقة أخرى للتعامل مع الذاكرة وهي الطريقة المباشرة حيث يحتوى الأمر نفسه على عنوان مكان الذاكرة المراد التعامل معه ، وإن ذلك مثلاً يقول لصديقك محمد يأخى يا محمد خذ هذه الرسالة وأنت مسافر إلى الرياض وأعطها إلى أخي أحمد ولا تنسى أن عنوان أخي أحمد مكتوب على الرسالة هذه المرة . الأمران الذين يقومان بهذه المهمة من قائمة أوامر الشريحة 8085 هما الأمران LDA و STA كما يلى :

4-2-4 الأمران LDA و STA

الأمر STA يعني خزن محتويات المركم STore Accumulator والأمر الثاني LDA يعني حمل المركم Load Accumulator والصورة العامة للأمرتين هي :

STA addr

محتويات المسجل A ————— العنوان (addr)

LDA addr

محتويات العنوان (addr) ————— المسجل A

والصورة المستعرضية للأمر STA هي :

32

البait ذات القيمة الصغرى من العنوان Addr

البait ذات القيمة العظمى من العنوان Addr

والصورة المستعرضية للأمر LDA هي :

3A

البait ذات القيمة الصغرى من العنوان Addr

البait ذات القيمة العظمى من العنوان Addr

الأمر الأول STA سيقوم بتخزين محتويات مسجل التراكم A في مكان الذاكرة الذي عنوانه في (الإثنين بait) التاليتين لشفرة الأمر نفسه وأما الأمر LDA فإنه يقوم بتحميل مسجل التراكم A بمحظيات مكان الذاكرة الذي عنوانه في (الإثنين بait) التاليتين لشفرة الأمر نفسه . هناك ملاحظتان هامتان على هذين الأمرين : الأولى هي أن هذين الأمرين لا يتعاملان إلا مع مسجل التراكم A فقط ، فأنتم مثلاً إن أردت أن تخزن محتويات المسجل B في أي مكان في الذاكرة بهذه الطريقة المباشرة فلا بد وأن تنقل محتويات المسجل B إلى المسجل A أولاً ثم تستخدم الأمر STA ، وكذلك الحال إن أردت أن تحمل أي مسجل ول يكن C بمحظيات أي مكان في الذاكرة بالطريقة المباشرة فعليك باستخدام الأمر LDA الذي يضع محتويات الذاكرة في مسجل التراكم ثم تقوم أنت بنقل هذه المحتويات من مسجل التراكم إلى أي مسجل آخر ول يكن C كما ذكرنا . الملاحظة الثانية هي أنه طالما أن هذين الأمرين يحتويان على عنوان فلا بد وأن يتكون كل منهما من ثلاثة بaitات ، واحدة هي شفرة الأمر ، أي 32 في حالة الأمر STA و 3A

فى حالة الأمر LDA وأما (الإثنين بait) الأخرى فتحتوى العنوان المراد التعامل معه كما فى الصورة العامة للأمرین . تذكر أن أى عنوان يتكون دائمًا من 16 بت أى 2 بait .

مثال 7-4

E000 E001 E002 STA E100
E003 E004 E005 LDA E101

الأمر الأول سيخزن محتويات المسجل A فى المكان E100 فى الذاكرة والأمر الثاني سيحمل المسجل A بمحتويات المكان E101 من الذاكرة ، لاحظ أنه فى ذاكرة البرنامج كل أمر من الأمرین يشغل ثلث بaitات .

القرار باستخدام أى واحدة من الطريقتين (المباشرة أو غير المباشرة) فى عملية البرمجة يتوقف على التطبيق الذى يستخدم فيه هذا البرنامج وسترجىء هذا الموضوع قليلا إلى أن ندرس بعض الأوامر الأخرى وعندها سنوضح أن هناك برامج يفضل فيها استخدام الطريقة المباشرة وأخرى لا بد فيها من استخدام الطريقة غير المباشرة .

5-2 الأمان SHLD و LHLD

معناهما load H,L Direct أى حمل المسجلين L,H مباشرة ، و Store H,L Direct أى خزن المسجلين L,H مباشرة وكل الأمرين كما نرى تأثيره عكس الآخر . الصورة العامة لهذين الأمرین هي :

LHLD addr

محتويات (addr) ← H ← (addr+1) ، محتويات (addr)

SHLD addr

محتويات (addr) ← L ← (addr+1) ، محتويات (addr)

الصورة المستعرية للأمر LHLD هي :

2A

البait ذات القيمة الصغرى للعنوان addr

البait ذات القيمة العظمى للعنوان addr

الصورة المستعرية للأمر SHLD هي :

22

البait ذات القيمة الصغرى للعنوان addr

البait ذات القيمة العظمى للعنوان addr

حيث يقوم الأمر الأول LHLD بتحميل المسجلين H و L بمحتويات العنوان addr والذى يليه فى الذاكرة ، أى أن محتويات العنوان addr تذهب إلى المسجل L ومحتويات العنوان الذى يليه addr+1 تذهب إلى المسجل H . الأمر SHLD

يقوم بالعملية العكسية للأمر LHLD حيث يقوم بتخزين محتويات المسجلين H و L في العنوان addr والذى يليه بحيث تذهب محتويات المسجل L إلى العنوان addr ومحتويات المسجل H إلى العنوان $addr+1$.

مثال 8-4

- افترض الوضع التالي في المسجلين HL ومكاني الذاكرة E100 و E101 :

H	L
89	76

E100	FF
E101	FF

بعد تنفيذ الأمر SHLD E100 سيصبح الوضع كما يلى :

H	L
89	76

E100	76
E101	89

- افترض الوضع التالي في المسجلين HL ومكاني الذاكرة E100 و E101 :

H	L
89	76

E100	3A
E101	B5

بعد تنفيذ الأمر LHLD E100 سيصبح الوضع كالتالى :

H	L
B5	3A

E100	3A
E101	B5

إلى هنا سنكتفى بهذه المجموعة من أوامر الإنتقال ولمزيد من المعرفة بباقي أوامر الإنتقال انظر الأشكال الموجودة في آخر هذا الفصل والتي تحتوى على نبذة مختصرة عن كل أمر .

3-4 تمارين

- اكتب برنامجا يضع الأرقام السبعية 1 إلى A في الذاكرة E201 إلى E20A.

2. اكتب برنامجا يضع الرقم 77 في المسجل D والرقم B4 في المسجل C ثم يقوم باستبدال محتويات كل من المسجلين ، أى أن محتويات المسجل D تذهب إلى المسجل C ومحتويات المسجل C تذهب إلى المسجل D دون أن تفقد أيا من محتويات المسجلين ، هذه العملية تسمى Swapping بمعنى استبدال أو مقايضة .
3. نفذ عملية الاستبدال السابقة في المسألة رقم 2 ولكن هذه المرة على محتويات البايت E100 والبايت E101 .
4. اكتب برنامجا للإزاحة الدورانية كالموجود في مثل 4-4 ولكن هذه المرة على محتويات الذاكرة E100 و E101 و E102 و E103 و E104 و E105 .
- ملاحظات :** يجب عمل مخطط سير لكل برنامج ثم ترجم هذه الخريطة إلى برنامج بلغة الأسsembli مع الشفرات الثنائية والستعرية وتحديد العناوين لكل أمور وذاكرة البرنامج وذاكرة البيانات لكل برنامج ، أى أن كل برنامج يحل بنفس طريقة حل المثال رقم 4-4 وحتى تكتمل الفائدة بالذات عند هذا المستوى يجب أن تحمل هذه البرامج بالشفرات الثنائية أو لا ثم الشفرات الستعرية ثم شفرات الأسsembli .

4-4 مجموعة أوامر الحساب Arithmatic Instructions

ADD
SUB
ADI
SUI
ADC
SBB
INR
INX
DCR
DCX

شكل (4-4) مجموعة أوامر الحساب

سندرس في هذا الجزء بعض الأوامر التي تقوم بإجراء العمليات الحسابية الأولية وهي الجمع والطرح . كما علمنا من قبل فإن مسجل التراكم لابد وأن يكون طرفا في أي عملية من هذه العمليات كما أن نتيجة هذه العملية سواء كانت جمعا أو طرحا تكون دائما موجودة في مسجل التراكم A . هناك أيضا خاصية مهمة

في هذه المجموعة من الأوامر غير موجودة في الأوامر الأخرى وهي أن أوامر الحساب (ومثلها أيضاً أوامر المنطق) عندما يتم تنفيذ أي أمر منها فإن جميع الأعلام الموجودة في مسجل الحالة SR تتأثر بنتيجة هذه العملية الحسابية أو المنطقية. راجع مسجل الحالة ومحطوياته ومتى يكون أي علم من الأعلام يساوى صفرًا أو واحداً وذلك في الفصل الثاني حيث أن جميع الأعلام الموجودة في مسجل الحالة في المعالج 8085 هي نفسها تماماً التي شرحت في هذا الجزء (4-2). شكل (4-4) يبين مجموعة الأوامر التي سندرسها هنا حيث في جميع الأمثلة التي سنستعين بها في هذا الجزء سنكتفى باستخدام شفرات الأسمبلية فقط على أساس أنه تم التدريب الكافي على الشفرات الثنائية والستعشرية مع مجموعة أوامر الانتقال ومن يريد الاستزادة في التدريب يمكنه الإستمرار في ذلك مستعيناً بالشفرات الستعشرية التي سنذكرها مع الصورة العامة لكل أمر وكذلك في الأشكال الموجودة في نهاية هذا الفصل.

4-4-1 الأوامر ADD و SUB

حيث ADD بمعنى إجمع و SUB هي اختصار الكلمة SUBtract بمعنى اطرح، والصورة العامة لأمر الجمع ADD هي :

ADD reg

$A \leftarrow A + reg$

حيث سيقوم هذا الأمر بجمع محتويات المسجل reg أو مكان الذاكرة M الذي عنوانه في المسجلين HL على محتويات مسجل التراكم A وستوضع نتيجة الجمع في المسجل A مع التأثير على جميع الأعلام . الشفرة الثنائية العامة لهذا الأمر هي 10000xxx حيث تستبدل xxx بشفرة المسجل الذي سيتم جمعه مع مسجل التراكم . كمثال على ذلك الأمر ADD B ستكون شفرته الثنائية هي 10000000 وشفرته الستعشرية هي 80H حيث استبدلت xxx بشفرة المسجل B وهي 000 .

الصورة العامة لأمر الطرح SUB هي :

SUB reg

$A \leftarrow A - reg$

يقوم هذا الأمر بطرح محتويات المسجل reg أو مكان الذاكرة M الذي عنوانه في المسجلين HL من محتويات مسجل التراكم وسوف توضع نتيجة الطرح في مسجل التراكم مع التأثير على جميع الأعلام . الشفرة الثنائية العامة لهذا الأمر هي 10010xxx حيث تستبدل xxx بشفرة المسجل المطلوب طرح محتوياته من مسجل التراكم . كمثال على ذلك الأمر SUB M ستكون شفرته الثنائية هي 10010110 وشفرته الستعشرية هي 96H .

مثال 9-4

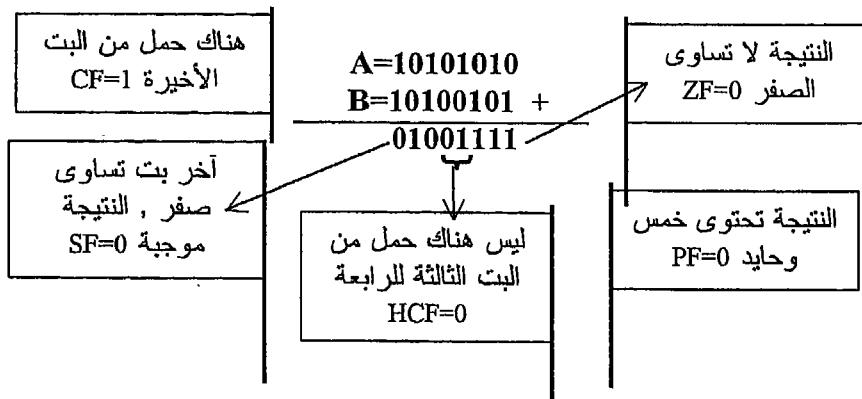
افترض المحتويات التالية للمسجلين A و B :



بعد تنفيذ الأمر ADD B ستصبح المحتويات كالتالي :



ولكي نرى كيف تتأثر الأعلام بنتيجة هذه العملية سنجري عملية الجمع على الشفرات الثنائية لكل من الرقمين كالتالى :



الآن افترض أنتانفذنا أمر الطرح SUB B على المحتويات الأولى للمسجلين A و B فإنه بعد تنفيذ هذا الأمر ستصبح محتويات المسجلين كالتالى :



ولكي نرى كيف تمت عملية الطرح وكيف تأثرت الأعلام سنجرى عملية الطرح على الشفرات الثنائية لمحطويات المسجلين A و B . كما نعلم فإن عملية الطرح الثنائى يتم تحويلها إلى عملية جمع حيث سنجمع محتويات المسجل A مع المتمم الثنائى لمحطويات المسجل B (انظر الملحق الأول فى نهاية الكتاب لمراجعة عمليات الجمع والطرح الثنائى). المتمم الثنائى لمحطويات المسجل B(10100101) هو 11011011 وبذلك تصبح عملية الطرح عملية جمع كالتالى :

$$\begin{array}{r}
 A = 10101010 \\
 01011011 + \text{ المتمم الثنائى لمحطويات المسجل B} \\
 \hline
 00000101
 \end{array}$$

بالنظر لهذه النتيجة ستكون الأعلام كالتالى :

- طالما أن البت الأولى على الأقل لا تساوى صفرًا فالنتيجة لا تساوى صفرًا ويكون علم الصفر يساوى صفرًا أي $ZF=0$.
- تحتوى النتيجة على عدد زوجي من الوحайд (اثنين) ، بذلك سيكون علم الباريتى يساوى واحداً أى $PF=1$.
- هناك حمل من البت الثالثة إلى البت الرابعة لذلك فعلم الحمل النصفى يكون واحداً $HCF=1$.
- آخر بت (رقم 7) تساوى صفرًا لذلك فالنتيجة موجبة وعلم الإشارة يكون دائماً مساوياً لمحطويات آخر بت إذن $SF=0$.
- المفروض أنه في عملية الطرح يهمنا أن نعرف إذا كان هناك استلاف أم لا لأنه في عملية الطرح لن يكون هناك حمل . بما أن عملية الطرح قد حولت إلى عملية جمع لذلك فإنه إذا كان هناك حمل في عملية الجمع فإن ذلك يعني أنه لن يكون هناك استلاف في عملية الطرح ويكون العلم $CF=0$ وهى الحالة التى نحن بصددها الآن والعكس صحيح إذا لم يكن هناك حمل في عملية الجمع .

4-4-2 الأمران ADI و SUI

الأمر الأول اختصاراً Add Immediate أي اجمع المعلومة الفورية أو الثابت ، والأمر الثنائى SUI اختصاراً Subtract Immediate والتي تعنى أيضاً اطرح المعلومة الفورية أو الثابت ، وقد أشرنا سابقاً إلى أن الحرف I في أي أمر يعني التعامل مع معلومة فورية أو ثابت يعطى في الأمر نفسه . الصورة العامة للأمر ADI هي :

ADI data8
 $A \leftarrow A + data8$

والشفرة السبعية العامة لهذا الأمر هي :

C6
 data8

حيث يقوم هذا الأمر بجمع محتويات البايت الثانية في الأمر مع محتويات مسجل التراكم A ويوضع النتيجة في المسجل A وعلى ضوء هذه النتيجة تتأثر جميع الأعلام . لاحظ أن هذا الأمر يتكون دائماً من اثنين بايت ، واحدة هي شفرة الأمر والثانية هي البايت أو الرقم المطلوب جمعه مع المسجل A .

الصورة العامة للأمر SUI هي :

SUI data8
 $A \leftarrow A - data8$

والشفرة المستعشرية لهذا الأمر هي :

D6
data8

حيث يقوم هذا الأمر بطرح محتويات البايت الثانية في الأمر من محتويات مسجل التراكم A ويوضع النتيجة في المسجل A وعلى ضوء هذه النتيجة تتأثر جميع الأعلام . لاحظ أن هذا الأمر أيضاً يتكون من اثنين بايت .

مثال 10-4

المطلوب هو جمع الثابت أو المعلومة الفورية 05 على محتويات المسجلات D, C, B . شكل (4-5) يبين مخطط السير وشفرات الأسsembli لهذا البرنامج . ملاحظة مهمة في هذا البرنامج هي أنه لكي نجمع الثابت مع أي مسجل فإننا نحضر أو ننقل محتويات المسجل أولاً إلى مسجل التراكم A ثم نجمع الثابت مع المسجل A وبالطبع فإن النتيجة تكون في المسجل A فنقوم بنقلها إلى المسجل الآخر ثانية . وهل يوجد حل آخر؟!

3-4-4 الأمران ADC و SBB

الأمر الأول يعني Add with Carry أي اجمع مع الحمل ، والثاني يعني ADC أو اطرح مع الاستلاف والصورة العامة للأمر ADC هي :

ADC reg
 $A \leftarrow A + CY + reg$

حيث يقوم هذا الأمر بجمع محتويات المسجل reg مع محتويات علم الحمل CY (لاحظ أن علم الحمل يكون إما صفرأ أو واحد) مع محتويات مسجل التراكم A والنتيجة توضع بالطبع في المسجل A . الشفرة الثانية لهذا الأمر هي :

10001xxx

حيث تستبدل ال xxx بشفرة المسجل المراد التعامل معه .

الصورة العامة للأمر SBB هي :

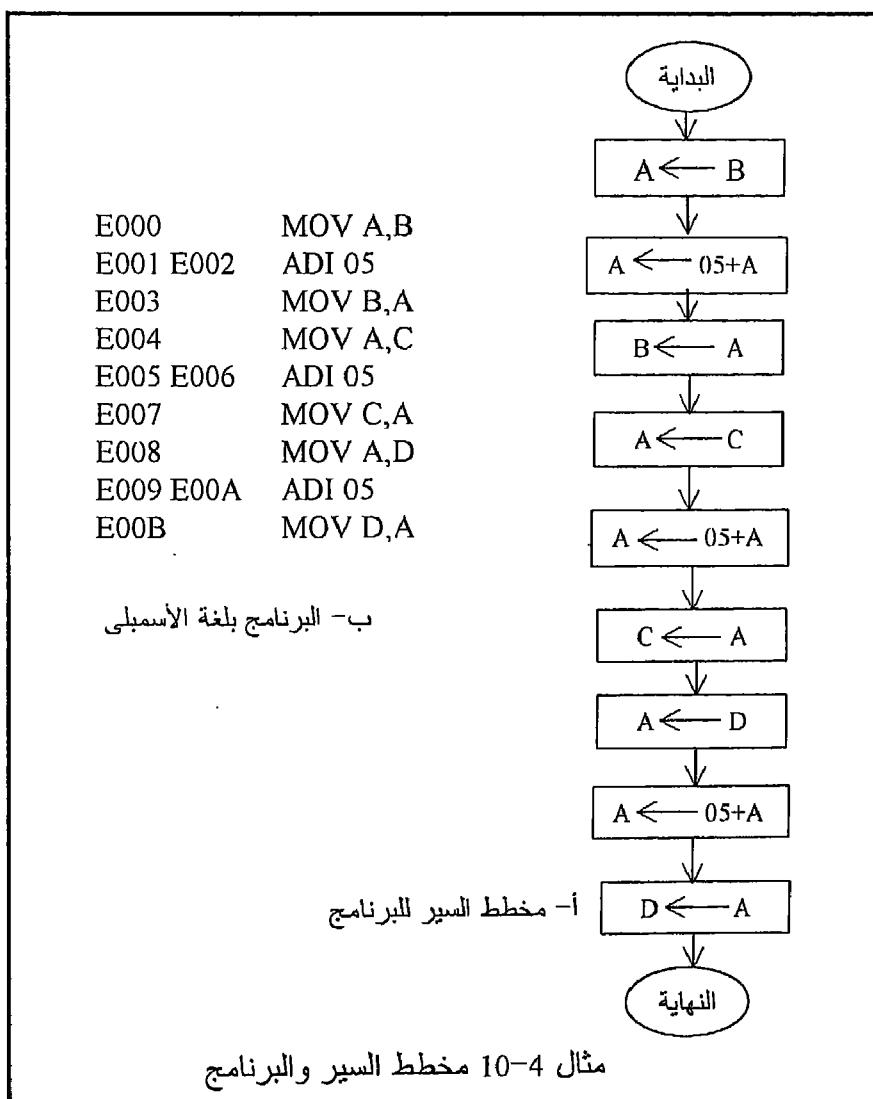
SBB reg

$A \leftarrow A - CY - reg$

حيث يطرح هذا الأمر من محتويات مسجل التراكم A محتويات المسجل reg مع ملاحظة أن المسجل A يكون هو المطروح منه دائمًا في جميع أوامر الطرح . الشفرة الثنائية لهذا الأمر هي :

10011xxx

حيث تستبدل ال xxx بشفرة المسجل المراد التعامل معه .



مثال 11-4

المطلوب جمع الرقمين 23F9H و 9A35H ووضع نتيجة الجمع في أماكن الذاكرة E100 و E101 و E102 . المشكلة في هذين الرقمين أن كلاً منها يشغل (اثنين بait) ونحن نعلم أن جميع أوامر الجمع تتعامل مع بايت واحدة فقط فما الحل ؟ الحل لهذه المشكلة ممكن بأن نضع الرقم الأول مثلاً في المسجلين B و C بحيث يحتوي المسجل B البايت ذات القيمة العظمى من الرقم وهي 23 ويحتوى المسجل C البايت ذات القيمة الصغرى وهي F9 وبنفس الطريقة نضع الرقم الثانى فى المسجلين D و E بحيث يحتوى المسجل D البايت 9A ويحتوى المسجل E البايت 35 . بعد ذلك سنجمع البايت ذات القيمة الصغرى في كل من الرقمين أى المسجل C زائد المسجل E باستخدام الأمر ADD وسينتج عن ذلك نتيجة وحمل ، النتيجة تخزنها في الذاكرة E100 ، وبعد ذلك نستخدم الأمر ADC لجمع البايت ذات القيمة العظمى في الرقمين مع محتويات علم الحمل وهي الحمل الناتج من عملية الجمع الماضية وسينتج عن ذلك نتيجة تخزن في الذاكرة E101 وحمل ، هذا الحمل يخزن في الذاكرة E102 كجزء من النتيجة . شكل (4-6) يبين رسمياً توضيحاً للحل وخريطة التدفق والبرنامج لهذا المثال .

قبل أن نترك الأمر ADC يجب أن نفهم جيداً متى يكون من الضروري استخدام الأمر ADC ؟ ومتى يكون من الضروري عدم استخدامه ؟ فمثلاً في المثال السابق (11-4) كان من الضروري عدم استخدام الأمر ADC في عملية الجمع الأولى (E + C) ولكن يجب استخدام الأمر ADD خوفاً من أن يكون علم الحمل CY به واحد من أي عملية سابقة ونحن لا ندرى فيجمع مع عملية الجمع الأولى وتكون النتيجة خطأ . أما في عملية الجمع الثانية (D + B + CY) فإنه لابد من استخدام الأمر ADC لأننا نريد أن نأخذ قيمة علم الحمل CY في الاعتبار .

4-4-4 الأمران INR و DCR

الأمر الأول يعني Increment أي زد المحتويات بمقدار واحد ، و Decrement أي انقص المحتويات بمقدار واحد ، والصورة العامة للأمر INR هي :

INR reg

reg \leftarrow 1 + reg

حيث يقوم هذا الأمر بجمع واحد على محتويات المسجل reg أو مكان الذاكرة M الذي عنوانه في HL . الصورة الثانية لهذا الأمر هي :

00xxx100

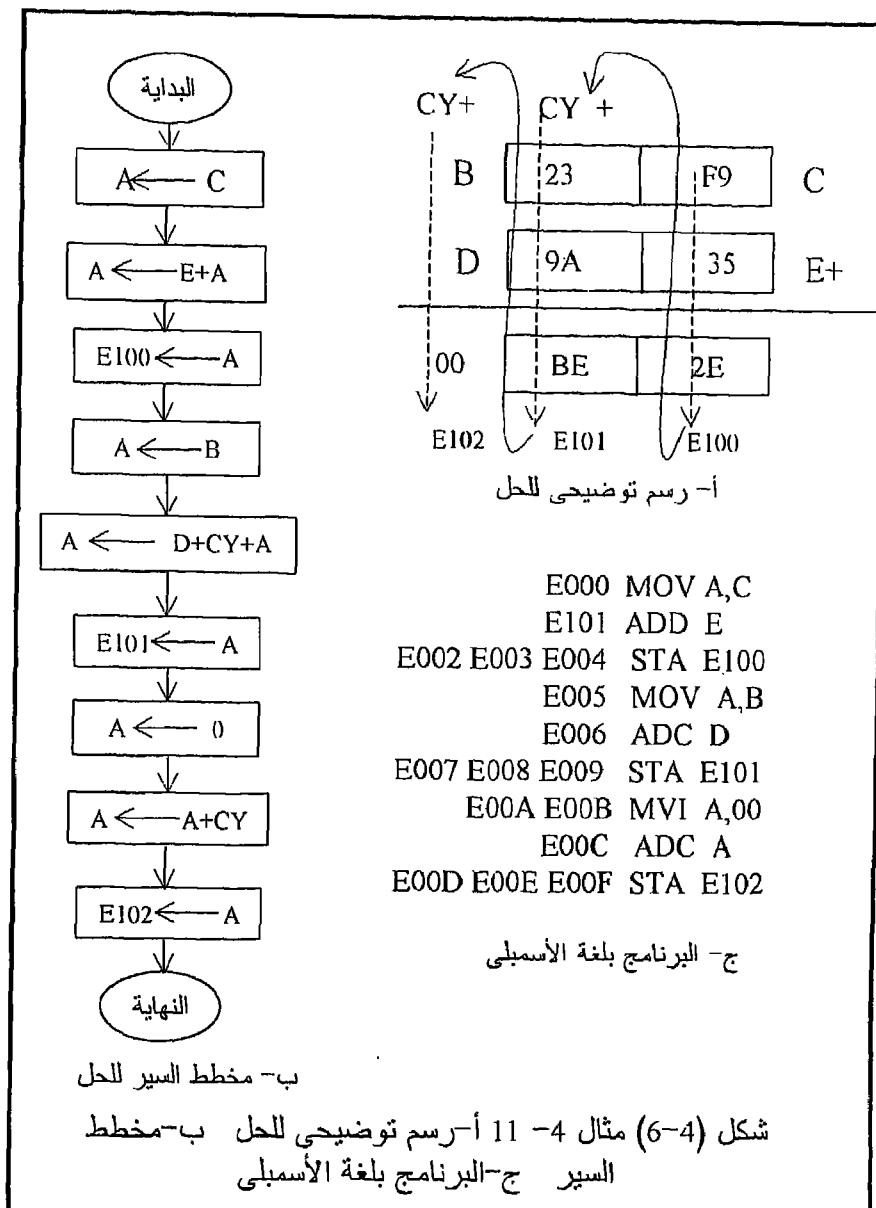
حيث xxx تستبدل بشفرة المسجل المراد التعامل معه .
الصورة العامة للأمر DCR هي :

DCR reg

$\text{reg} \leftarrow \text{reg} - 1$

حيث يقوم هذا الأمر بطرح واحد من محتويات المسجل reg أو مكان الذاكرة M الذي عنوانه في HL . الصورة التالية لهذا الأمر هي :
00xxx101

حيث xxx تستبدل بشفرة المسجل المراد التعامل معه .



5-4-4 الأمران INX و DCX

الأمران INX و DCX هما مرادفات للأمرين INR و DCR ولكنهما يستخدمان مع أزواج مسجلات ، فالأمر INX يجمع واحداً على محتويات زوج من المسجلات والصورة العامة الأسمبلى والثانوية له هي :

INX rp

00xx0011

rp ← 1 + rp

حيث xx تستبدل بشفرة زوج المسجلات المراد التعامل معه . وأما الأمر DCX فإنه يطرح واحداً من محتويات زوج من المسجلات وصورته العامة (الأسمبلى والثانوية) هي :

DCX rp

00xx1011

rp ← rp - 1

حيث xx تستبدل بشفرة زوج المسجلات المراد التعامل معه .

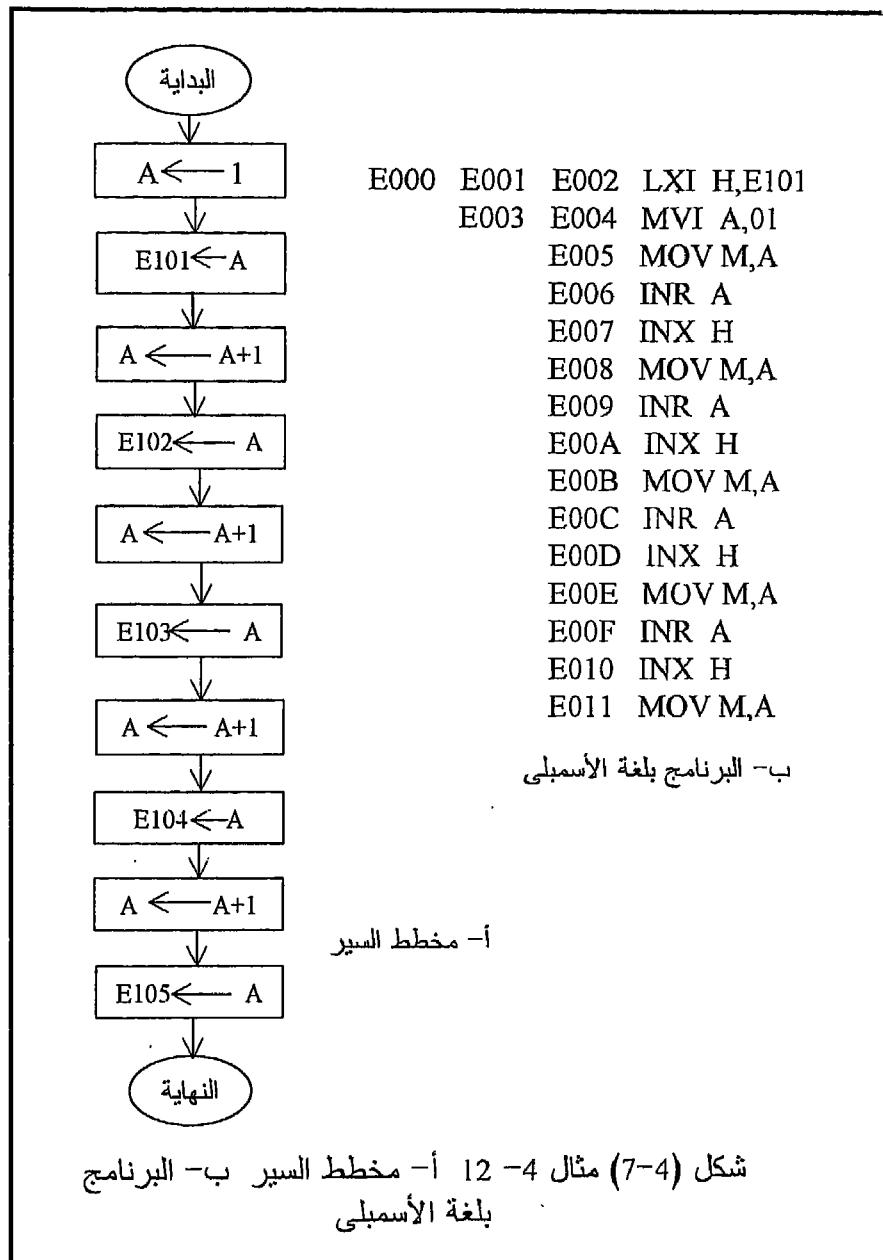
مثال 12-4

المطلوب تخزين الأرقام 01, 02, 03, 04, 05 في أماكن الذاكرة التي عناوينها E101, E102, E103, E104, E105 على التوالي . شكل (7-4) بيّن خريطة التدفق البرنامج لحل هذه المسألة . لاحظ استخدام الأمر INR لزيادة واحد على محتويات المسجل A والأمر INX لزيادة واحد على محتويات زوج المسجلات HL في هذا المثال . نلاحظ من شكل (7-4) أننا لكي نخزن خمسة أرقام في الذاكرة كتبنا برماجا مكوناً من ثمانية عشرة بait (E000 حتى E011) بما العمل لو أننا نريد تخزين ألف رقم أو أكثر ، كم ستشغل من ذاكرة البرنامج...! هل هناك من حل لتخفيف عدد أوامر مثل هذه البرامج ؟ إن الحل لذلك هو استخدام أوامر القفز والحلقات وهو موضوع الجزء القادم بعد أن نعرض بعض التمارين كتطبيق على أوامر الحساب .

5-4 تمارين

1. اكتب برماجا يجمع محتويات أماكن الذاكرة E101, E102, E103, E104 مع ما يناظرها من الأماكن E10A, E10B, E10C, E10D, E10E ثم يضع النتيجة في الأماكن E110, E111, E112, E113, E114 .
2. أعد البرنامج السابق مستخدماً الطرح بدلاً من الجمع .

3. اكتب برمجا يجمع الرقم F3A56BH مع الرقم 78B6A9H ويضع النتيجة في بايتات الذاكرة E103, E102, E101, E100 .
4. أعد البرنامج السابق مستخدما الطرح بدلا من الجمع .



4-6 مجموعة أوامر القفز

Jump Instructions

القاعدة العامة أن المعالج يقوم بتنفيذ البرنامج حسب ترتيب الأوامر الموجودة فيه من أول البرنامج إلى نهايته ، ولقد كان حريصين في جميع الأمثلة السابقة على الحفاظ على هذه القاعدة ، ولكن هناك بعض التطبيقات التي تتطلب الخروج على هذه القاعدة كأن يتطلب منك مثلاً تنفيذ عملية معينة عدد معين من المرات أو عدد لا نهائي من المرات . فعندما يكون المعالج مثلاً مراقباً لدرجة الحرارة في عملية صناعية معينة فإن عليه أن يقرأ درجة الحرارة ويقارنها بدرجة حرارة مخزنة في الذاكرة كمراجع ، وإذا زادت الحرارة عن حد معين يقوم المعالج بضرب جرس إنذار مثلاً وإذا نقصت عن حد معين يشغل سخان لزيادتها ، مثل هذا البرنامج سيكون عبارة عن مجموعة من الأوامر التي تتفذ إلى ما لا نهاية طالما أن المعالج يراقب درجة الحرارة .

لقد أتاح المعالج هذه العملية بتوفير بعض الأوامر التي تمكنك كمبرمج من القفز بعملية التنفيذ من مكان آخر خلال البرنامج . هناك نوعان من القفز :

4-6-1 القفز غير المشروط unconditional jump

عند تنفيذ أي عملية قفز غير مشروط ينتقل المعالج بعملية التنفيذ إلى المكان الجديد دون أي قيد أو شرط ، وهناك أمر واحد فقط من أوامر المعالج 8085 يقوم بهذه العملية ، والصورة العامة لهذا الأمر هي :

JMP addr

حيث إنه عند تنفيذ هذا الأمر يوضع العنوان addr الذي سيتم القفز إليه في عدد البرنامج فيصبح الأمر الموجود عند هذا العنوان هو الأمر الذي عليه الدور في التنفيذ . لاحظ أن هذا الأمر يتكون من ثلاثة بaites واحدة هي شفرة الأمر واثنان للعنوان addr الذي سيتم القفز إليه .

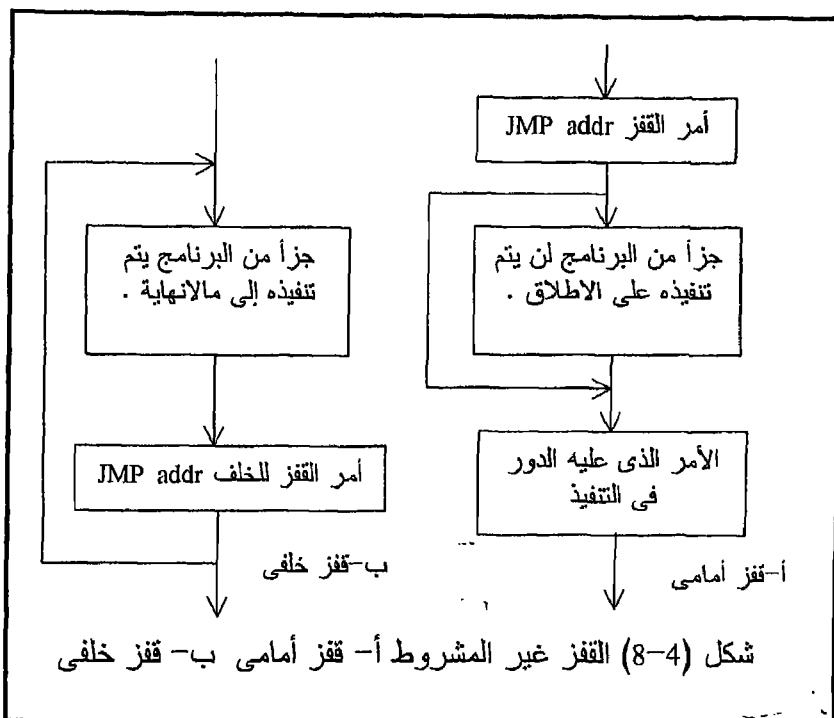
إن القفز باستخدام الأمر JMP addr قد يكون إلى الأمام في البرنامج وقد يكون إلى الخلف . إذا كان القفز إلى الأمام سيتخرج عن ذلك وجود جزء من البرنامج لن ينفذ على الإطلاق وهو الجزء الذي بين أمر القفز JMP والأمر الذي سيتم القفز إليه . أما إذا كان القفز إلى الخلف فإنه سيتخرج عن ذلك ما يسمى بالحلقة اللانهائية Infinite loop والتي سوف يستمر المعالج في تنفيذها إلى ما لا نهاية . شكل (4-8) يبين مخطط السير لعملية القفز غير المشروط الأمامي والخلفي .

4-6-2 القفز المشروط conditional jump

كما يوحى الإسم فإنه في هذا النوع من القفز لن يتم القفز إلا إذا تحقق شرط معين أما إذا لم يتحقق الشرط فإن البرنامج ينفذ في التتابع الطبيعي حيث سينفذ

الأمر الذى بعد أمر القفز مباشرة . إن شروط القفز توضع دائما على الأعلام التي في مسجل الحالة ، فيمكنك مثلا أن تجعل القفز مشروطا بأن تكون النتيجة صفرأ أو تجعل القفز مشروطا بأن تكون النتيجة سالبة ، وهكذا ، حيث أن هناك خمسة أعلام واحد منها وهو علم الحمل التصفى HC لا يستخدم كشرط فى عمليات القفز فإنه يتبقى أربعة أعلام يمكن أن تستخدم فى ثمانية من أوامر القفز المشروط كالالتلى :

JZ addr	اقفز إذا كانت النتيجة صفرأ
JNZ addr	اقفز إذا كانت النتيجة ليست صفرأ
JM addr	اقفز إذا كانت النتيجة سالبة
JP addr	اقفز إذا كانت النتيجة موجبة
JC addr	اقفز إذا كان هناك حمل
JNC addr	اقفز إذا لم يكن هناك حمل
JPO addr	اقفز إذا كانت الباريتى فردية
JPE addr	اقفز إذا كانت الباريتى زوجية



لاحظ أن عدد هذه الأوامر ثمانية ، إثنان منها لكل علم من الأعلام الأربع تمثل جميع الحالات التي يمكن أن يكون فيها هذا العلم (صفرأ أو واحدا) . إن جميع

هذه الأوامر لابد وأن تكون ثلاثة بait ، واحدة هي شفرة الأمر op code واثنتان للعنوان الذى سيتم القفز إليه إذا تحقق الشرط . لاحظ أن النتيجة التى نعنيها فى الأوامر السابقة هي آخر نتيجة تأثرت بها الأعلام ، ولذلك فإنه قبل أن نكتب أي أمر من أوامر القفز غير المشروط يجب أن ندرس جيداً هل الأمر السابق لأمر القفز المشروط يؤثر على الأعلام أم لا كما سيتضح من المثال التالى :

مثال 4-13

اكتب برنامجاً يقرأ محتويات بait الذاكرة E100 باستمرار (إلى ما لا نهاية) ثم يختبر هذه المحتويات بحيث إذا كانت صفرًا يضع 1 في المدخل B وإذا كانت سالبة يضع 2 في المدخل B وإذا كانت موجبة يضع 4 في نفس المدخل B . شكل (4-9) يبين خريطة التدفق والبرنامج لهذا المثال . البرنامج الموجود فى شكل (4-9) يعتبر تدريباً على معظم أفكار الحلقات ، فيه الحالات اللانهائية الناتجة عن الأمر JMP addr ، كما أن فيه القفز المشروط JP addr . كما أن فيه القفز إلى الأمام في البرنامج ، والقفز إلى الخلف أيضاً . لذلك يجب دراسة هذا البرنامج بدقة وإمعان .

هناك أوامر أخرى يتسبب عنها قفز بعملية تنفيذ البرنامج إلى أماكن أخرى وهي أوامر القفز إلى البرامج الفرعية والعودة منها وسوف نرجئ الدراسة التفصيلية للبرامج الفرعية إلى فصل آخر خاص بذلك .

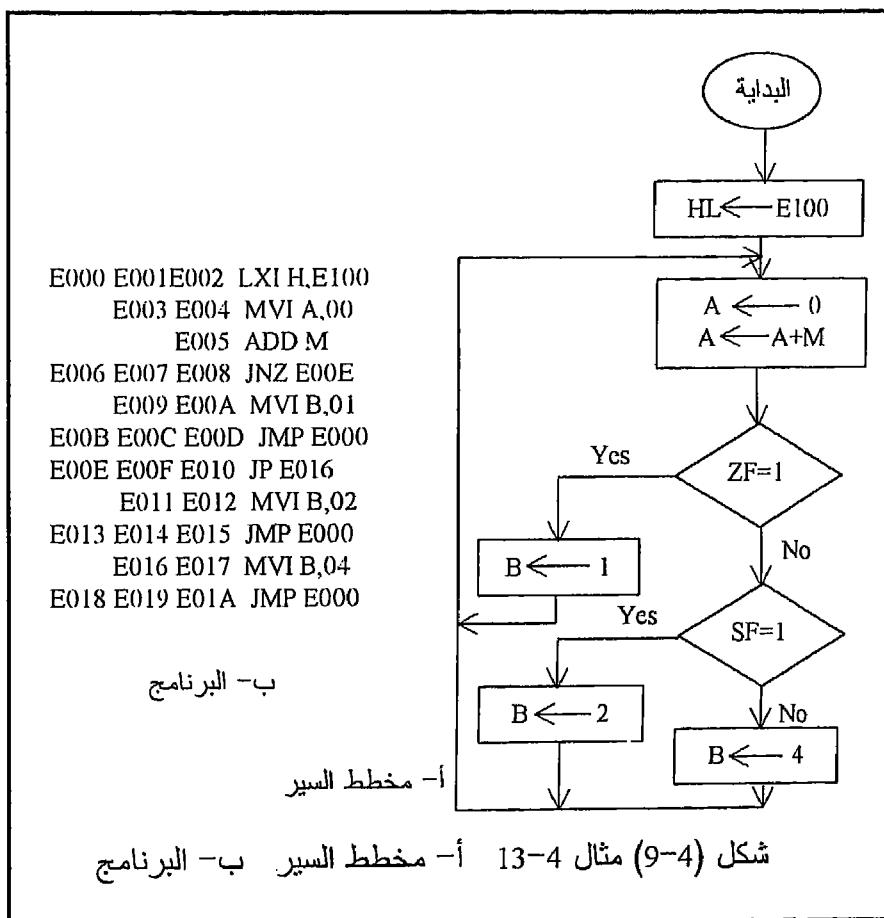
7-4 مهمة أخرى للأسمبلر

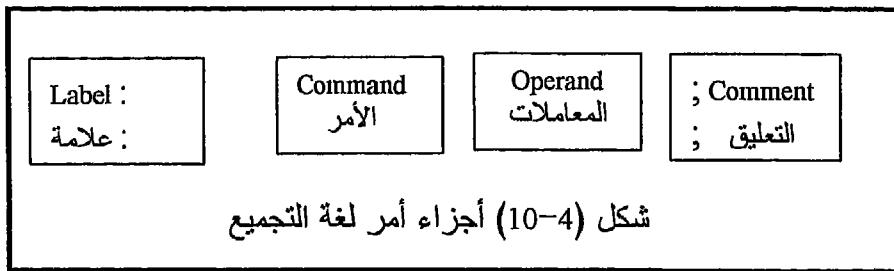
المهمة الوحيدة التي عرفناها حتى الآن للأسمبلر هي مهمة تحويل الشفرات الحرفية (الأسمبلر) إلى شفرات ثنائية أو لغة الماكينة ، ولكن لحسن الحظ فإن الأسمبلر قد تم تزويد بعض المهام الأخرى التي تريح المبرمج إلى درجة كبيرة والتي سنتعرف على واحدة منها في هذا الجزء . ينظر أي أسمبلر إلى الأمر الذي تكتبه له على أنه مكون من أجزاء أربعة كالموضحة في شكل (4-10) والتي سندرسها بالترتيب فيما يلى :

1-7-4 الأمر والمعاملات

لقد درسنا الكثير من أوامر لغة الأسمبلر والتي سنلخصها كلها في شكل واحد وفي ترتيب أبجدي في آخر هذا الفصل إن شاء الله . من هذه الأوامر MOV, ADD, JMP, وغيرها . كل أمر من هذه الأوامر لابد وأن يكون له معاملات وهذه المعاملات هي المسجلات أو أماكن الذاكرة التي سوف يؤثر عليها الأمر ،

فمثلاً الأمر MOV A,B معاملاته هي المسجلين A, B والأمر C معاملاته هي المسجل C والأمر M معاملاته هي بait الذاكرة التي عنوانها في المسجلين HL وهكذا . هناك أوامر قليلة ليست لها معاملات على الإطلاق ومنها الأمر NOP والذي معناه لا تعمل شيئاً No Operation كما أن عدد المعاملات في أي أمر لا يزيد عن اثنين بأى حال . إن الأمر يجب أن يفصل عن معاملاته بمسافة واحدة على الأقل وإن زاد عدد المسافات عن واحدة فلن يضر ذلك في شيء حيث أن الأسمبلر يهملاها ، أما إذا لم توجد مسافة واحدة على الأقل بين الأمر ومعاملاته فإن الأسمبلر سيعطي خطأ على ذلك ولن يقبل منك هذا الأمر . إذا كان الأمر له معاملان كالأمر MOV A,B فإن المعاملين يجب أن يفصل بينهما بفاصلة (,) وإذا لم يحدث ذلك فإن الأسمبلر يعطي رسالة خطأ على ذلك .





Comment 2-7-2 التعليق

حتى تجعل برنامجك مفهوماً ومن السهل قراءته وتتبعه بالنسبة للأخرين فإنه يجب عليك أن تكتب بعض التعليقات البسيطة بجانب كل أمر . لذلك فإن الأسمبلر يعطيك فرصة أن تكتب أي شيء تريده في نهاية الأمر على أن تفصل بين الأمر والتعليق بفاصلة منقوطة يفهم منها الأسمبلر أن كل المكتوب بعدها يعتبر تعليقاً ولا يدخل ضمن الأمر . إذا كان التعليق الذي ستكتبه سيأخذ أكثر من خط واحد فإن كل خط يجب أن يبدأ بفاصلة منقوطة . المثال التالي سيوضح أهمية كتابة التعليق في نهاية الأمر .

مثال 4.14

```

; Multiplication of two numbers
E000 E001 MVI A,00 ; Put 0 into A, Clean accumulator
E002 ADD B          ; Addition of A and B, result goes to A
E003 DCR C          ; Decrement C by 1
E004 E005 E006 JNZ E002 ;Go to add B to itself one more time
                         ;as long as C not equal to 0
E007 E008 E009 STA E100 ; store result in location E100
; -----

```

هذا البرنامج يضرب الرقم الموجود في المسجل B في الرقم الموجود في المسجل C ويضع النتيجة في بait الذاكرة رقم E100 . بما أنه ليس هناك أمر من أوامر الشريحة 8085 يقوم بإجراء عملية الضرب لذلك كان لابد أن نكتب هذا البرنامج لإجراء عملية الضرب عن طريق تكرار الجمع حيث يجمع البرنامج محتويات المسجل B مع نفسه عدد من المرات يساوى الرقم الموجود في المسجل C . ستجد أن التعليقات المكتوبة بجانب الأوامر تدل على ذلك . التعليق الموجود في السطر الأول مثلا يقول "ضع صبرا في A ، تنظيف المركم " وهذا ضروري حتى لا نجمع أي رقم قد يكون في المسجل A قبل البدء في عملية جمع المسجل B مع نفسه . التعليق في السطر الثاني يقول "جمع A+B ، النتيجة في A " لاحظ

أن هذا الأمر هو بداية حلقة يتم القفز إليها من السطر الرابع وفي المرة الأولى من دخول هذه الحلقة سيجمع المعالج المسجل $A=0$ مع محتويات المسجل B فيصبح $A=B$. التعليق في السطر الثالث يقول "انقص محتويات C بمقدار واحد". التعليق في السطر الرابع يقول "ارجع لجمع المسجل B مع نفسه مرة أخرى طالما أن C لا يساوى صفرًا ، النتيجة في A ". التعليق في السطر الخامس يقول "خزن النتيجة في البایت E100". التعليق في السطر السادس وهو الخط المنقطع تم وضعه فقط للدلالة على نهاية البرنامج وهذا أحياناً يكون مهماً جداً في عملية تنظيم كتابة البرنامج خاصة إذا كان البرنامج مكوناً من أكثر من جزء حيث يمكن الفصل بين الأجزاء المختلفة عن طريق مثل هذه الخطوط التي يعرفها الأسمايل على أنها تعليقات . بالطبع يجب أن تكون التعليقات باللغة الإنجليزية ، إلا إذا كان محرر الكلمات الذي تستخدمه في كتابة البرنامج سيسمح لك باستخدام اللغة العربية في التعليقات .

3-7-4 العلامة Label

في جميع البرامج التي كتبناها حتى الآن كنا حريصين تماماً على أن نكتب عناوين البایتات التي يشغلها أي أمر من أوامر البرنامج ، وكانت هذه العملية ضرورية بالذات عند التعامل مع أوامر القفز التي تحتاج فيها لمعرفة عنوان الأمر الذي سنقفز إليه مثل الأمر الرابع في المثال السابق (مثال ضرب الرقمين 14-4 وهو JNZ E002 ، فلولا أننا كنا نكتب عناوين البایتات التي عندها الأوامر لما حدثنا أن القفز يجب أن يكون إلى الأمر الثاني وهو ADD B E002 . في الحقيقة إن مهمة تتبع العنوانين لجميع أوامر البرنامج تعتبر مهمة صعبة للمبرمج وبالذات إذا كان البرنامج يحتوى على الكثير من أوامر القفز ، وتعتبر مهمة سهلة للأسمبلر يستطيع القيام بها في نفس الوقت . فطالما أن الأسمايل يعرف جيداً عدد البایتات التي يتكون منها كل أمر فلم لا يتولى هو عملية العنونة لأوامر البرنامج على أن يعطيه المبرمج فقط عنوان أول أمر في البرنامج ، وما على المبرمج في هذه الحالة إلا الكتابة فقط بالشفرات الأسمايل ، هنا يبرز سؤال مهم : كيف سنحدد للأسمبلر الأوامر التي من الممكن أن يتم القفز إليها ؟ إن ذلك يتم عن طريق العلامات Labels التي نضعها قبل الأمر المراد القفز إليه على أن نستخدم نفس العلامة في أمر القفز نفسه ، إننا سنبعد كتابة البرنامج الموجود في المثال 14-4 مرة أخرى دون استخدام عناوين للأوامر وباستخدام العلامات كما في المثال 4-15 .

مثال 4-15

MVI A,00 ; Put 0 into accumulator

```

HERE: ADD B      ; A+B into A
      DCR C      ; Decrement C by 1
      JNZ HERE    ; Jump to add B to itself one more
                  ; time as long as C not equal to 0
      STA E100    ; Store result in E100
;

```

لاحظ أننا وضعنا العلامة وهي كلمة **HERE**: كعلامة عند الأمر الذي سيتم القفز إليه وهو الأمر الثاني في البرنامج السابق ، واستخدمنا نفس العلامة **HERE** في الأمر **JNZ HERE** كى يتم القفز إلى الأمر رقم 2 . من البديهي أنه لابد وأن يكون هناك تطابق تام بين العلامة في الأمر الذي سيتم القفز إليه والعلامة في أمر القفز نفسه بحيث إذا لم يوجد هذا التطابق فإن الأسمبلر سيعطى خطأ . أيضاً لابد وأن تنتهي العلامة الموجودة في أول الأمر الذي سيتم القفز إليه بالحرف (:) ليميز بها الأسمبلر بين نهاية العلامة وبداية الأمر. كما أن عدد أحرف العلامة يجب ألا يزيد عن ثمانية أحرف لا تبدأ برقم .

البرنامج الموجود في المثال 4-14 مكتوب في الذاكرة ابتداء من البايت E000 ، افترض مثلاً أننا لأى سبب نريد نقل البرنامج بحيث يبدأ من البايت E050 بدلًا من E000 . في هذه الحالة لابد وأن نعيد فحص البرنامج بدقة ونغير العنوانين الموجودة في جميع أوامر القفز في البرنامج ، فمثلاً الأمر **JNZ E002** في المثال 4-14 سيصبح **JNZ E052** لو أننا بدأنا البرنامج من البايت E050 ، وهكذا تخيل لو أن البرنامج به العديد من أوامر القفز فإنه لا شك ستكون عملية إعادة عنونة الأوامر من الصعوبة بمكان وبالذات في البرامج الطويلة . إن استخدام العلامات **Labels** كما في مثال 4-15 يريح المبرمج تماماً من عملية تتبع العنوانين في أوامر القفز حتى لو تغير مكان البرنامج لأنك تعطي الأسمبلر عنوان بداية البرنامج فقط وهو ينسب جميع العلامات إلى عنوان البداية وهذه في الحقيقة فائدة عظيمة يوفرها الأسمبلر للمبرمج . لذلك فإن جميع البرامج التي سنعرضها فيما بعد سنكتبها باستخدام لغة الأسمبلر والعلامات دون الحرص على كتابة عنوانين الأوامر لأننا بذلك تكون قد عبرنا مرحلة لا يأس بها في لغة الأسمبلر .

8-4 أوامر الإدخال والإخراج Input Output instructions

إلى الآن رأينا كيف نبرمج شريحة المعالج وكيف نحرر المعلمات داخلها من مسجل إلى مسجل آخر أو من مسجل إلى الذاكرة أو العكس ولكن لم نعرف حتى الآن كيف نظهر معلومة على شاشة عرض مثلاً أياً كان نوع هذه الشاشة ، أو

كيف ندخل معلومة إلى المعالج من خلال لوحة مفاتيح على سبيل المثال . إن شاشة العرض ولوحة المفاتيح يعتبران مثالان من ضمن العديد من الأمثلة التي تحتاج لعمليات الإخراج والإدخال . فمثلاً حينما يستخدم المعالج في التحكم في أي متغير في أي عملية صناعية ول يكن مثل درجة الحرارة فإنه لابد من إدخال درجة الحرارة إلى المعالج بعد تهيئتها ووضعها في الصورة المناسبة لذلك ، وكذلك إذا أراد المعالج رفع درجة الحرارة أو ضرب جرس إنذار فإنه لابد وأن يخرج إشارة معينة تؤخذ وتهيأ في الصورة المناسبة للجهاز الذي ستذهب إليه سواء كان سخاناً أو جرساً أو غيره . جميع عمليات الإدخال والإخراج تتم من خلال ما يسمى ببوابات الإدخال والإخراج والتعامل مع هذه البوابات دائمًا ينقسم إلى قسمين : قسم خاص بالبناء الإلكتروني لهذه البوابات وكيفية توصيلها مع المعالج وهذا القسم سندرسه بالتفصيل في فصل قادم إن شاء الله ، والقسم الآخر هو كيفية برمجة المعالج للتعامل مع هذه البوابات وهو موضوع دراستنا في هذا الجزء حيث سندرس الأوامر الخاصة بهذه العملية وسنفترض في دراستنا في هذا الجزء أن القارئ لديه على الأقل بوابة إدخال واحدة وبواية إخراج واحدة موصولة على الميكروكمبيوتر الذي يتدرّب عليه ويكتب عليه برامجه .

4-8-1 أمر الإدخال IN وأمر الإخراج OUT

وهما اختصار لكلمتى Input يعني أدخل و Output يعني أخرج والصورة العامة للأمر IN هي :

IN no.

A البوابة رقم no. ← المسجل

حيث سيقوم هذا الأمر بإدخال المعلومة الموجودة على بوابة الإدخال والتي رقمها no. في الأمر نفسه إلى مسجل التراكم A . هذا الأمر يتكون من اثنين بaitt واحده هي شفرة الأمر IN والثانية هي رقم البوابة المراد التعامل معها ، ولذلك فإنه يمكن التعامل مع $2^8 = 256$ بوابة إدخال تبدأ من البوابة رقم 00H إلى البوابة رقم FFH . انظر شفرة الأمر IN السبع عشرية في جدول الأوامر في نهاية هذا الفصل .

الصورة العامة لأمر الإخراج OUT هي :

OUT no.

no. ← البوابة رقم A ← المسجل

حيث سيقوم هذا الأمر بإخراج المعلومة الموجودة في مسجل التراكم A إلى بوابة الإخراج التي رقمها no. هذا الأمر يتكون من اثنين بaitt واحده هي شفرة الأمر OUT والثانية هي رقم البوابة المراد التعامل معها ، ولذلك فإنه يمكن التعامل مع $2^8 = 256$ بوابة إخراج تبدأ أيضاً من البوابة رقم 00H إلى البوابة رقم FFH . انظر شفرة الأمر OUT السبع عشرية في جدول الأوامر في نهاية هذا الفصل .

لاحظ أن كلا من بوابة الإدخال وبوابة الإخراج تتكون من 8 بิตات مثلاً في ذلك مثل أي مسجل يتكون من 8 بิตات ولذلك فإن أكبر رقم يمكن أن يكون على أي بوابة هو الرقم 255 . تذكر أيضاً أن أي تعامل مع أي بوابة باستخدام هذين الأمرتين لابد وأن يكون من خلال المركم A ، فإذا أردت أن تخرج محتويات المسجل C مثلاً إلى بوابة الإخراج رقم 05 فإنه لابد وأن تنقل هذه المحتويات إلى المسجل A أولاً ثم تنفذ أمر الإخراج كما يلى :

انقل محتويات C إلى A : MOV A,C

أخرج على بوابة الإخراج رقم 05 : OUT 05

بعد تنفيذ الأمرين السابقين ستري محتويات المسجل C على بوابة الإخراج رقم 05 . أما إذا أردنا أن ندخل محتويات بوابة الإدخال رقم 00 ونخزنها في بايت الذاكرة رقم E100 فإننا ننفذ الأمرين التاليين على سبيل المثال :

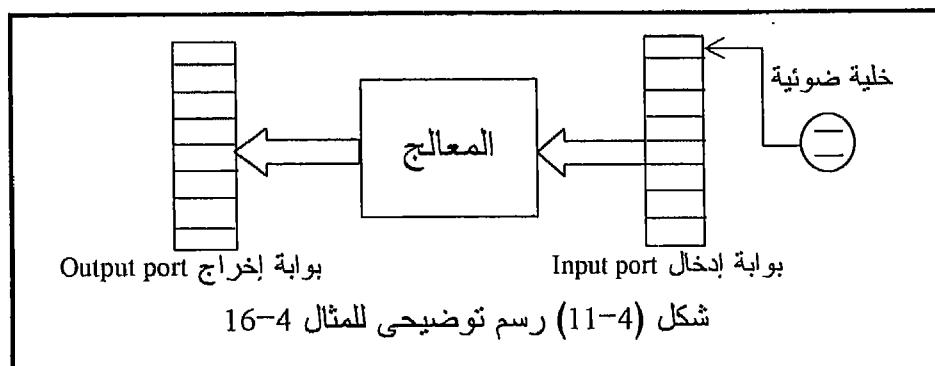
أدخل محتويات بوابة الإدخال رقم 00 إلى المركم : IN 00

خزن محتويات المركم في البايت E100 : STA E100

بعد تنفيذ هذين الأمرين فإنه إذا كانت محتويات بوابة الإدخال رقم 00 تسلوى 55 مثلاً فإن هذا الرقم (55) ستجده في البايت E100 .

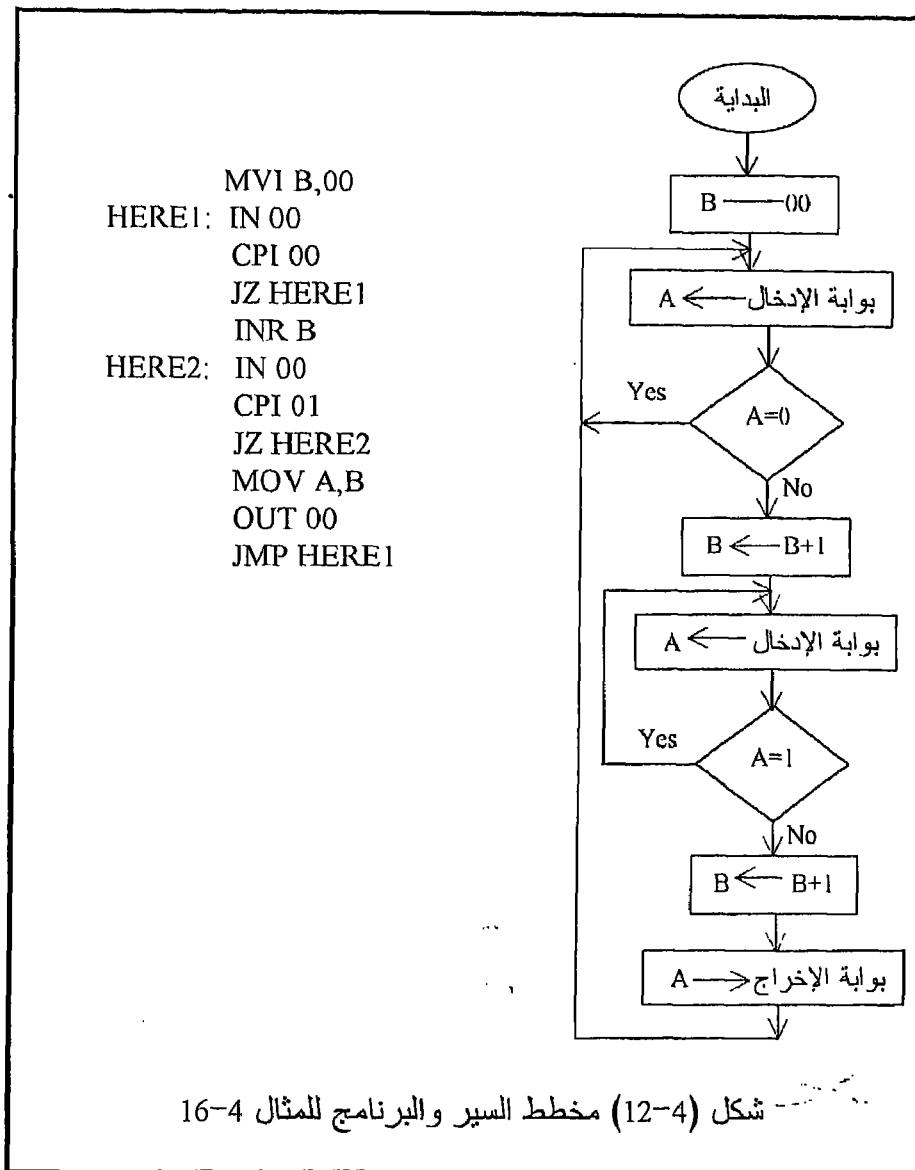
مثال 16-4

إفترض أن لدينا خط إنتاج في أحد المصانع تعبر عليه المنتجات وفي أثناء العبور فإن كل منتج يقطع خلية ضوئية فتعطى نبضة كهربائية على خرجها . خرج هذه الخلية موصى على البت رقم 0 في بوابة الإدخال رقم 00H . المطلوب هو كتابة برنامج يعد هذه المنتجات ويخرج العدد على بوابة الإخراج رقم 00H أيضاً . شكل (11-4) يبين رسمًا توضيحيًا لهذه العملية وشكل (12-4) يبين مخطط السير والبرنامج لهذا المثال .



لاحظ وجود الأمر CPI وهو أحد أوامر الحساب التي لم نشرحها وتركناها للقارئ لمراجعة لها من قوائم الأوامر في نهاية الفصل ، ولقد استخدمنا هذا الأمر

لمقارنة محتويات المسجل A القادمة من بوابة الإدخال بالقيمة 00 أو لا ، وطالما أن هذه المحتويات تساوى 00 فإن ذلك يعني عدم مرور أى منتج على خط الإنتاج ويقوم المعالج بتكرار عملية القراءة حيث يقفز إلى HERE1 كما فى البرنامج .



شكل (12-4) مخطط السير والبرنامج للمثال 16-4

عندما يمر أى منتج س تكون محتويات بوابة الإدخال مختلفة عن الصفر ولن يرجع المعالج إلى العلامة HERE1 ثانية ولكنه سيزيد محتويات المسجل B بمقدار

واحد حيث B تعتبر عداداً للمنتجات المارة على الخط ، ثم بعد ذلك سيقوم المعالج بقراءة بوابة الإدخال مرة أخرى للتأكد من أن المنتج قد مر من على الخط وذلك بمقارنة محتويات بوابة الإدخال بالقيمة 01 وطالما أنها تساوى 01 يقفل إلى العلامة HERE2 حيث لا يعمل شيئاً سوى قراءة البوابة . عندما تنزل البوابة إلى 00 مرة أخرى فإن ذلك يدل على أن المنتج قد مر على الخط فيقوم المعالج بإخراج قيمة العداد B على بوابة الإخراج ويدعوه إلى العلامة HERE1: مرة أخرى لمواصلة البرنامج .

4-9 مجموعة أوامر المنطق Logic Instruction Set

العمليات المنطقية التي يستطيع المعالج القيام بها هي العمليات XOR, NOT, OR, AND وسنكتفي هنا بعرض الصورة العامة لهذه الأوامر على أن يقوم القارئ بمراجعة هذه الأوامر في القوائم الموجودة في آخر الفصل لمعرفة الشفرات الثنائية والزمن الذي يأخذه كل أمر لكي يتم تنفيذه . كما ذكرنا سابقاً فإن العمليات المنطقية مثلها مثل العمليات الحسابية لابد وأن يكون المركم طرفاً فيها كما أن النتيجة لابد وأن توضع في المركم أيضاً . جميع العمليات المنطقية تؤثر على الأعلام ، انظر قوائم الأوامر في آخر الفصل لتدرك ذلك ولترى متى تكون هذه الأوامر مكونة من بait واحدة ومتى تكون مكونة من 2 bait .
الصورة العامة للأمر AND هي :

ANA reg

A AND reg ← مسجل A

حيث سيقوم المعالج بإجراء عملية AND على محتويات المسجل reg مع محتويات مسجل التراكم A ونتيجة العملية توضع في المركم . بنفس الطريقة يمكننا كتابة الصورة العامة للأمرين OR و XOR كما يلى :

ORA reg

XRA reg

حيث سيقوم الأمر الأول بإجراء عملية OR على محتويات المسجل reg مع محتويات المسجل A ويوضع النتيجة في المسجل A . وأما الأمر الثاني فسيجري عملية XOR على محتويات المسجل reg مع محتويات المركم ويوضع النتيجة في المركم . جميع العمليات المنطقية الثلاث السابقة يمكن أن تجرى على معلومة فورية أو ثابت وفي هذه الحالة فإن الصورة العامة لهذه الأوامر ستكون :

ANI data8

ORI data8

XRI data8

حيث data8 هو الثابت أو المعلومة الفورية المراد إجراء العملية المنطقية عليها .
لاحظ أن هذه الأوامر في هذه الحالة سيكون كل منها مكونا من 2 بايت ، واحدة هي شفرة الأمر والثانية هي قيمة الثابت data8 .

مثال 17-4

افتراض أن محتويات المسجل A تساوى FFH ومحتويات المسجل B تساوى F0H والمطلوب إجراء عمليات AND و OR على كل من المسجلين A و B .

أولاً : عملية AND

$$A=11111111$$

$$B=11110000$$

النتيجة بعد إجراء الأمر AND هي :

$$\begin{array}{r} 11110000 \\ \hline 11110000 \end{array}$$

أى أن محتويات المركم ستتغير إلى FOH . جميع الأعلام ستتأثر بهذه النتيجة ما عدا علمي الحمل والحمل النصفى حيث يكونان صفراء دائما فى جميع العمليات المنطقية لأنه لا يحدث لا حمل ولا حمل نصفى مع أى عملية منطقية .

ثانياً : عملية OR

$$A=11111111$$

$$B=11110000$$

النتيجة بعد إجراء الأمر OR هي :

$$\begin{array}{r} 11111111 \\ \hline 11111111 \end{array}$$

أى أن محتويات المركم ستظل FFH وستتأثر الأعلام بنفس الطريقة التى ذكرناها مع عملية AND .

ثالثاً : عملية XOR

$$A=11111111$$

$$B=11110000$$

النتيجة بعد إجراء الأمر XOR هي :

$$\begin{array}{r} 00001111 \\ \hline 00001111 \end{array}$$

أى أن محتويات المركم ستصبح 0FH وستتأثر جميع الأعلام بنفس الطريقة السابقة .

بذلك تكون قد أنهينا دراسة المجموعات الأساسية في أوامر المعالج 8085 على أنها لم نذكر بالطبع جميع الأوامر داخل كل مجموعة ولكننا بذلك تكون قد عرفنا الغالبية العظمى من الأوامر وما تبقى منها فمن السهل معرفته من جداول الأوامر الموجودة في آخر هذا الفصل كما أن هناك أبواب خاصة ستعرض لمجموعات معينة من الأوامر مثل الأوامر الخاصة بالبرامج الفرعية والأوامر الخاصة بالمقاطعة حيث سيفرد فصل خاص لكل منها إن شاء الله نتيجة لأهميتها .

4-10 كيفية الاتصال بالذاكرة Memory addressing

أى معلومة من حيث المكان إما أن تكون موجودة داخل المعالج نفسه فى أى مسجل من مسجلاته وفي هذه الحالة فإنه يمكن التعامل معها بسهولة وتحريكها من مكان لأخر داخل المعالج فى زمن أقل ، وإما أن تكون موجودة فى الذاكرة ويريد المعالج قرأتها ، أى إحضارها من الذاكرة ووضعها فى أى مسجل من مسجلاته ، أو كتابتها أى نقلها إلى الذاكرة ، وذلك يتطلب من المعالج أن يحدد عنوان المكان أو البايت فى الذاكرة التى ستتم معها عملية القراءة أو الكتابة . طريقة تحديد عنوان البايت من الذاكرة المراد التعامل معها هي المقصد دراسته فى هذا الجزء . يجب أن نتذكر جيدا أن جميع شرائح المعالجات التى نتعامل معها إلى الآن لها مسار عناوين به 16 بتا ولذلك فإننا عندما سنكتب أى عنوان فى النظام المستعشرى فلا بد أن نكتبه من 4 خانات حيث كل خانة فى النظام المستعشرى تمثل 4 بتات فى النظام الثنائى ، فمثلا العنوان 1111000010100001 فى النظام الثنائى هو F0A1H في النظام المستعشرى وغالبا نضع الحرف H للدلالة على أن الرقم مكتوبا في النظام المستعشرى . هناك طريقتان يحدد بهما المعالج 8085 عنوان المكان أو البايت في الذاكرة المراد التعامل معه وسنبين هاتين الطريقتين فيما يلى :

4-10-1 الطريقة المباشرة Direct method

في هذه الطريقة فإن الأمر نفسه يحتوى عنوان البايت المراد التعامل معها ، ولذلك فإن كل الأوامر التي تقع تحت هذا الصنف تتكون من ثلاثة بايتات ، واحدة من هذه البايتات هي شفرة الأمر operation code واختصارا نكتب op code والإثنين بايت التالية هي عنوان المكان في الذاكرة المراد التعامل معه . تذكر أن البايت تتكون من 8 بتات وأنه دائما تكون البايت الأولى من بايتات العنوان هي البايت ذات القيمة الصغرى Low significant byte . بالنسبة للمعالج 8085 هناك الأمران LDA addr و STA addr كامثلة على الأوامر التي تتعامل بهذه الطريقة . كما رأينا سابقا فإن الأمر LDA addr يقوم بتحميل المسجل A (المركم) بمحطيات المكان في الذاكرة الذي عنوانه موجود في الأمر نفسه وقد رمزنا له بالرمز addr . أما الأمر STA addr فإنه يفعل العكس حيث يقوم بتخزين محطيات المسجل A في المكان الذي عنوانه addr في الذاكرة .

4-10-2 الطريقة غير المباشرة Indirect method

في هذه الطريقة يوضع عنوان البايت المراد التعامل معها في المسجلين H و L بحيث يحتوى المسجل H على البايت ذات القيمة العظمى من العنوان ويحتوى

المسجل L على البايت ذات القيمة الصغرى منه . مثل هذه الأوامر تكون دائمًا من بايت واحدة حيث يرمز لهذا المكان في الذاكرة بالرمز M وتأخذ الشفرة 110 كما رأينا من خلال تعاملنا مع الأوامر سابقا . فمثلاً الأمر MOV A,M معناه انتقال محتويات بايت الذاكرة التي عنوانها في زوج المسجلات HL إلى مسجل التراكم .

السؤال الذي يتبرد إلى الذهن هنا أي الطريقتين يفضل في الاستخدام ، الطريقة المباشرة أم غير المباشرة ؟ الإجابة عن هذا السؤال تتوقف على البرنامج أو على المشكلة التي نبرمجها . فإذا كان البرنامج يتعامل مع الذاكرة باستمرار وبالذات إذا كان التعامل مع أماكن متعددة في الذاكرة فإن الطريقة غير المباشرة لا شك تكون الأفضل لأنها ستتوفر الكثير من طول البرنامج لأن المسجلين HL في هذه الحالة سيكونان عبارة عن مؤشر إلى بايت الذاكرة التي تستخدمها وكلما أردت التعامل مع بايت جديد تزيد محتويات المسجلين HL بواحد كما رأينا في المثال 7-4 . أما إذا كنت ستتعامل مع الذاكرة لمرة واحدة فإنه ليس هناك أي داع لاستخدام الطريقة غير المباشرة ولكن يفضل في هذه الحالة الطريقة المباشرة . الأشكال التالية تحتوى مجموعة أوامر المعالج 8085 مقسمة أولاً إلى مجموعات كما أشرنا سابقاً بحيث يحتوى كل شكل الأوامر الخاصة بمجموعة معينة والشفرة العشرية وعدد نصوص التزامن التي يحتاجها كل أمر لكي يتم إحضاره من الذاكرة وتفيذه . شكل (4-18) يبين قائمة أوامر الشريحة 8085 مرتبة ترتيباً أبجدياً مع ملخص أو نبذة عن وظيفة كل أمر وكذلك الأعلام التي تتأثر بكل أمر .

MOV	A	B	C	D	E	H	L	M
,A	7F	47	4F	57	5F	67	6F	77
,B	78	40	48	50	58	60	68	70
,C	79	41	49	51	59	61	69	71
,D	7A	42	4A	52	5A	62	6A	72
,E	7B	43	4B	53	5B	63	6B	73
.H	7C	44	4C	54	5C	64	6C	74
.L	7D	45	4D	55	5D	65	6D	75
.M	7E	46	4E	56	5E	66	6E	

MVI	A	B	C	D	E	H	L	M
	31EXX	06XX	0EEXX	16XX	1EEXX	26XX	21EXX	36XX

LDA (13)	STA (13)
3AXXXX	32XXXX

LDAX B (7)	LDAX D (7)	STAX B (7)	STAX D (7)
0A	1A	02	12

IN (10)	OUT (10)	LHLD (16)	SHLD (16)
DBXX	D3XX	2AXXXX	22XXXX

XCHG (4)	XTHL (16)	SPHL (6)
EB	E3	F9

	LXI (10)	POP (10)	PUSH (10)
B	01XXXX	C1	C5
D	11XXXX	D1	D5
H	21XXXX	E1	E5
SP	31XXXX		
PSW		F1	F5

- جميع الأوامر التي على الصورة MOV reg,M و MOV M,reg تأخذ 7 نبضات تزامن والأوامر التي على الصورة MOV reg,reg تأخذ 4 نبضات تزامن .
 - XX تعنى معلومة 8 بناres (data8) و XXXX تعنى عنوان أو معلومة 16 بناres .
 - جميع الأوامر التي على الصورة MVI reg,XX تأخذ 7 نبضات تزامن حتى يتم احضارها وتنفيذها والأمر MVI M,XX يأخذ 10 نبضات .
 - الأرقام التي بين القوسين للأوامر الأخرى تدل على عدد النبضات التي يأخذها الأمر .
- شكل (4-13) مجموعة أوامر الانتقال للمعالج 8085

	INR	DCR	ADD	SUB	ADC	SBB
A	3C	3D	87	97	8F	9F
B	04	05	80	90	88	98
C	0C	0D	81	91	89	99
D	14	15	82	92	8A	9A
E	1C	1D	83	93	8B	9B
H	24	25	84	94	8C	9C
L	2C	2D	85	95	8D	9D
M	34	35	86	96	8E	9E

	INX	DCX	DAD
B	03	0B	09
D	13	1B	19
H	23	2B	29
SP	33	3B	39

ADI(7)	SUI(7)	ACI(7)	SBI(7)	DAA(4)
C6xx	D6xx	CExx	DExx	27

- الأوامر **SBB M** ، **ADC M** ، **SUB M** ، **ADD M** كلها تأخذ 7 نبضات لكي يتم إحضارها وتنفيذها .
- الأوامر **INR M DCR M** تأخذ 10 نبضات .
- جميع أوامر الحساب الأخرى تحتاج إلى 4 نبضات .
- الأرقام التي بين القوسين لبعض الأوامر تدل على عدد النبضات اللازمة لإحضار وتنفيذ هذا الأمر .

شكل (14-4) مجموعة أوامر الحساب

	ANA	ORA	XRA	CMP
A	A7	B7	AF	BF
B	B0	A0	A8	B8
C	A1	B1	A9	B9
D	A2	B2	AA	BA
E	A3	B3	AB	BB
H	A4	B4	AC	BC
L	A5	B5	AD	BD
M	A6	B6	AE	BE

ANI	ORI	XRI	CPI	CMA
E6xx	F6xx	EExx	FExx	2F

- جميع أوامر المتنطق تأخذ 4 نبضات ما عدا الأوامر التي تتعامل مع ذاكرة M والأوامر ANI ORI XRI CPI فتأخذ 7 نبضات لكي يتم إحضارها وتنفيذها.

شكل (15-4) مجموعة أوامر المتنطق

JMP	JC	JNC	JZ	JNZ	JM	JP	JPE	JPO
C3 XXXX	DA XXXX	D2 XXXX	CA XXXX	C2 XXXX	FA XXXX	F2 XXXX	E8 XXXX	E2 XXXX

- هذه الأوامر إذا تم لها القفز تأخذ 10 نبضات وإذا لم يتم القفز تأخذ 7 نبضات لكي يتم إحضارها وتنفيذها .

CALL	CC	CNC	CZ	CNZ	CM	CP	CPE	CP O
CD XXXX	DC XXXX	D4 XXXX	CC XXXX	C4 XXXX	FC XXXX	F4 XXXX	EC XXXX	E4 XXXX

- هذه الأوامر تأخذ 18 نبضة إذا تم لها القفز وإذا لم يتم القفز تأخذ 9 نبضات .

RET	RC	RNC	RZ	RNZ	RM	RP	RPE	RPO
C9	D8	D0	C8	C0	F8	F0	E8	E0

- هذه الأوامر تأخذ 12 نبضة إذا تم القفز و 6 إذا لم يتم ما عدا الأمر RET فإنه يأخذ 10 نبضات .

RST 7	RST 6	RST 5	RST 4	RST 3	RST 2	RST 1	RST 0
FF	F7	EF	E7	DF	D7	CF	C7

- هذه الأوامر تأخذ 12 نبضة لكي يتم إحضارها وتنفيذها .

PCHL
E9

- هذا الأمر يحتاج إلى 6 نبضات لكي يتم إحضاره وتنفيذه .

شكل (16-4) مجموعة أوامر القفز

RLC	RRC	RAL	RAR	CMC	STC	EI	DI	HLT	NOP
07	0F	17	1F	3F	37	FB	F3	76	00

RIM	SIM
20	30

- جميع هذه الأوامر تحتاج إلى 4 نبضات لكي يتم إحضارها وتنفيذها .

شكل (17-4) أوامر أخرى

الأسمبلى	الشفرة	الأعلام المتأثرة	وظيفة الأمر
ACI const	11001110 data8	Z S P CY HC	ثابت + علم الحمل + A ← A
ADD reg	10000sss	Z S P CY HC	مسجل + A ← A +
ADC reg	10001sss	Z S P CY HC	مسجل + علم الحمل + A ← A
ADI const	11000110 data8	Z S P CY HC	ثابت + A ← A +
ANA reg	10100sss	Z S P 0 0	مسجل + A ← A AND
ANI const	11100110 data8	Z S P 0 0	ثابت + A ← A AND
CALL addr	CD addr16		نداء غير مشروط على برنامج فرعى
CC addr	DC addr16		نداء برنامج فرعى مشروط بعلم الحمل = 1
CM addr	FC addr16		نداء برنامج فرعى مشروط بنتيجة سالبة
CMA	2F		اعكس محتويات المسجل A
CMC	3F		اعكس علم الحمل
reg CMP	10111sss	Z S P CY HC	قارن مسجل مع مسجل A . اطرح (A - المسجل A) لا يتأثر بالنتيجة .
CNC addr	D4 addr16		نداء برنامج فرعى مشروط بعلم الحمل = 0
CNZ addr	C4 addr16		نداء برنامج فرعى مشروط بعلم الصفر = 0
CP addr	F4 addr16		نداء برنامج فرعى مشروط بنتيجة موجبة
CPE addr	EC addr16		نداء برنامج فرعى مشروط بباريتى زوجية
CPI const	FE data8	Z S P CY HC	مقارنة ، (ثابت - A) والمسجل A لا يتأثر
CPO addr	E4 addr16		نداء برنامج فرعى مشروط بباريتى فردية
CZ addr	CC addr16		نداء برنامج فرعى مشروط بعلم الصفر = 1
DAA	27	Z S P CY HC	حول المركم إلى النظام العشري

DAD rp	00rp1001	CY	$HL \leftarrow HL + rp$
DCR reg	00ddd101	Z S P CY HC	انقص محتويات المسجل reg بمقدار 1
DCX rp	00rp1011		انقص محتويات المسجلين rp بمقدار 1
DI	F3		اهمل المقاطعة
EI	FB		اسمح بالمقاطعة
HLT	76		أوقف تنفيذ البرنامج
IN no.	DB Port no.		اقرأ بوابة الإدخال رقم. no.
INR reg	00ddd100	Z S P CY HC	اجمع 1 على محتويات المسجل reg
INX rp	00rp0011		اجمع 1 على محتويات المسجلين rp
JC addr	DA addr16		قفز مشروط بعلم الحمل = 1
JM addr	FA addr16		قفز مشروط بعلم الإشارة = 1
JMP addr	C3 addr16		قفز غير مشروط
JNC addr	D2 addr16		قفز مشروط بعلم الحمل = 0
JNZ addr	C2 addr16		قفز مشروط بعلم الصفر = 0
JP addr	F2 addr16		قفز مشروط بعلم الإشارة = 0
JPE addr	EA addr16		قفز مشروط بباريتي زوجية
JPO addr	E2 addr16		قفز مشروط بباريتي فردية
JZ addr	CA addr16		قفز مشروط بعلم الصفر = 1
LDA addr	3A addr16		محتويات عنوان \leftarrow المسجل A
LDAX rp	00rp1010		محتويات العنوان الموجود في الزوج A أو DE تنقل للمسجل BC
LHLD addr	2A addr16		محتويات العنوان addr والذى يليه HL \leftarrow المسجلين
LXI rp,const	00rp0001		ثابت من 16 بت \leftarrow زوج

	data16		المسجلات rp
MOVreg1,reg2	01dddsss		محطيات reg1 ← reg2
MVIreg,const	00ddd110 data8		ثابت من 8 بت → reg
NOP	00		لا تعمل شيء
ORA reg	10110sss	Z SP 0 0	المسجل reg مع المسجل A OR
ORI const	F6 data8	Z SP 0 0	ثابت من 8 بت مع المسجل A OR
OUT no.	D3 Port no.		المسجل A ← بوابة الإخراج رقم no.
PCHL	E9		قفز للعنوان الموجود في المسجلين HL
POP rp	11rp0001		محطيات قيمة المكذبة rp ←
PUSH rp	11rp0101		محطيات rp ← قيمة المكذبة
RAL	17	CY	دوران المركم للشمال من خلال علم الحمل
RAR	1F	CY	دوران المركم لليمين من خلال علم الحمل
RC	D8		عودة مشروطة بعلم الحمل = 1
RET	C9		عودة غير مشروطة
RLC	07	CY	دوران المركم للشمال ، آخر بت الى البت الأولى وإلى علم الحمل ولا يذهب علم الحمل لأول بت
RM	F8		عودة مشروطة بنتيجة مالية
RNC	D0		عودة مشروطة بعلم الحمل = 0
RNZ	C0		عودة مشروطة بعلم الصفر = 0
RP	F0		عودة مشروطة بنتيجة موجبة
RPE	E8		عودة مشروطة بباريتي زوجية
RRC	0F	CY	دوران المركم لليمين ، أول بت إلى آخر بت وإلى علم الحمل ولا يذهب علم الحمل لآخر بت
RST	11nnn111		ابداً من العنوان صفر
RZ	C8		عودة مشروطة بعلم الصفر = 1
RIM	20		اقرأ قناع المقاطعة

SBB reg	10010sss	Z S P C Y H C	طرح reg وعلم الحمل من المسجل A
SBI const	DE data8	Z S P C Y H C	اطرح ثابت وعلم الحمل من المسجل A
SHLD	22		خزن محتويات المسجلين HL في الذاكرة
SIM	30		ضم قناع المقاطعة
SPHL	F9		حمل مؤشر المكادسة من المسجلين HL
STA add	32 addr16		خزن محتويات المركم في العنوان addr
STAX rp	00rp0010		خزن المركم في العنوان الموجود في المسجلين rp (BC و DE فقط)
STC	37	- - - 1 -	اجعل علم الحمل = 1
SUB reg	10010sss	Z S P C Y H C	اطرح المسجل reg من المسجل A
SUI const	D6 data8	Z S P C Y H C	اطرح ثابت من المسجل A
XCHG	EB		بدل محتويات المسجلين DE مع HL
XRA reg	10101sss	Z S P 0 0	المسجل reg مع المركم XOR
XRI const	EE data8	Z S P 0 0	ثابت مع المركم XOR
XCHG	EB		بدل محتويات المسجلين HL مع مؤشر المكادسة

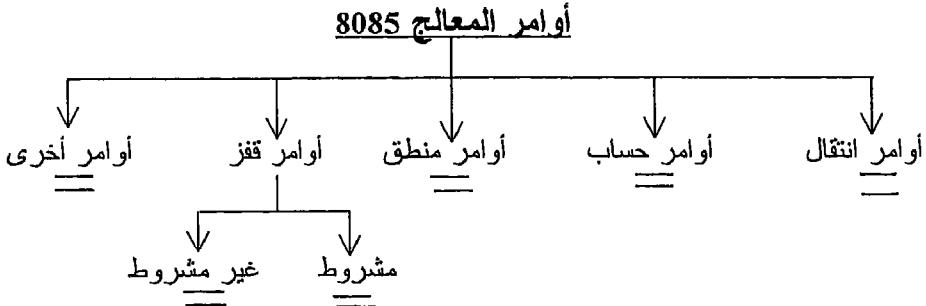
ملاحظات :

- اختصار الكلمة register وتعني مسجل 8 بت .
- اختصار الكلمة register pair وتعني زوجا من المسجلات .
- اختصار الكلمة constant وتعني ثابت أو معلومة فورية وأحيانا يكون هذا الثابت 8 ببات وأحيانا يكون 16 بتا على حسب الأمر إذا كان يتعامل مع مسجل واحد أو زوج من المسجلات .
- اختصار الكلمة address وتعنى عنوانا في الذاكرة ودائما يكون العنوان 16 بتا .
- Z تعنى علم الصفر S
- تعنى علم الإشارة P
- تعنى علم الباريتى HC
- CY تعنى علم الحمل
- تعنى علم الحمل الثنائي .

شكل (4.18) أوامر الشريحة 8085 مرتبة أبجديا

11-4 تمارين

1. أكمل الجدول التالي الخاص بأوامر الشريحة 8085 :



2. ما هي نتيجة تنفيذ البرنامج التالي :

E000 MVI A,05
 E002 MOV B,A
 E003 MOV C,B
 E004 MOV D,C
 E005 MOV E,D
 E006 MOV L,E
 E007 MOV H,L
 E008 MOV M,L

3. اقرأ البرنامج السابق وأجب عما يلى :

- محتويات مكان الذاكرة E000=.....
- محتويات مكان الذاكرة E001=.....
- محتويات مكان الذاكرة 0505=.....

.4

E000 LXI H,E100
 E003 MVI M,3E
 E005 INX H
 E006 MVI M,05
 E008 INX H
 E009 MVI M,47
 E00B INX H
 E00C MVI M,48

ما هي نتيجة تنفيذ البرنامج السابق ؟

5. على ضوء نتيجة تنفيذ البرنامج السابق ما هي نتيجة تنفيذ الشفرات الموجودة في الأماكن E100 إلى E103 ؟
6. هل تتأثر الأعلام بأوامر الانتقال ؟
7. اذكر الأعلام التي تتأثر بكل عملية من العمليات الحسابية والمنطقية ؟
8. إذا كانت محتويات المسجل A=F3H و محتويات المسجل B=A4H فاكتب محتويات المسجل A بعد تنفيذ كل أمر من الأوامر التالية على نفس المحتويات السابقة ووضح أيضاً كيف ستتأثر الأعلام بكل أمر :

ADD B
SUB B
SUB A
INC A
ANA B
ORA B
XRA B

9. ارسم خريطة تدفق للبرنامج التالي وما هي نتيجة تنفيذه :
- E000 MVI L,50H
E003 MVI H,E1H
E005 MOV M,A
E006 DCR L
E007 JNZ E005
10. ماذا يحدث لو كتبنا البرنامج السابق عند E100 بدلاً من E000 ؟
11. أعد كتابة البرنامج السابق مستخدماً العلامات Labels ؟ وما هي مميزات البرنامج مكتوباً بهذه الصورة ؟
12. كم عدد بوابات الإدخال التي يستطيع المعالج 8085 التعامل معها؟
13. كم عدد بوابات الإخراج التي يستطيع المعالج 8085 التعامل معها؟
14. على ماذا يتوقف هذا العدد ؟
15. هل هناك ما يمنع أن تكون بوابتي إدخال وإخراج لهما نفس الرقم . كمثال على ذلك OUT 05 و IN 05 ؟
16. OUT C,reg هذا أحد أوامر الإخراج للبروسبيسور Z80 والذي يعني إخراج محتويات المسجل reg على بوابة الإخراج التي رقمها في المسجل C فهل المعالج 8085 لديه ما يكفيه هذا الأمر ؟
17. هل تتأثر الأعلام بأوامر الإدخال والإخراج ؟
18. أكتب برنامجاً يقرأ محتويات البوابة 00 وإذا كانت هذه المحتويات زوجية يخزنها في الذاكرة ابتداءً من العنوان E100 وإذا كانت فردية يخرجها على البوابة 00 ؟

19. المعالج 8085 لديه طريقتان فقط لعنونة الذاكرة وهما العنونة المباشرة والعنونة غير المباشرة ، انكر الأوامر التي تستخدم مع كل من الطريقتين ؟
20. انكر متى تفضل استخدام العنونة المباشرة ومتى تفضل العنونة غير المباشرة ؟
21. المعالجان Z80 و MC6800 بهما طرق أخرى لعنونة الذاكرة والتي منها على سبيل المثال العنونة المفهرسة indexed addressing فهل هناك طرق أخرى لعنونة الذاكرة لدى المعالج 8085 ؟
22. اكتب برنامجاً يحسب عدد الوحدات الموجودة في محتويات المسجل A ، فمثلاً إذا كان $A=11110101$ فإن عدد الوحدات = 6 .
23. اكتب برنامج يحسب أكبر قيمة عدديّة في لبّيات في المدى العنوانى E200H إلى E250H .
24. اكتب برنامج يحسب عدد البّياتات التي تحتوي أصفراً والتي تحتوي أرقاماً موجبة والتي تحتوي أرقاماً سالبة في المدى العنوانى E100 إلى E150 .
25. اكتب برنامج يحسب عدد البّياتات التي تحتوي بيانات فردية والتي تحتوي بيانات زوجية في المدى العنوانى E100 إلى E150 .
26. المدى العنوانى E100 إلى E150 يحتوى بيانات لإشارة صوت ، احسب كم مرة عبرت إشارة الصوت الصفر .
27. اكتب برنامج يقرأ بوابة الإدخال رقم 00 ويختبر البّت الرابع فيّها ، فإذا كانت هذه البّت واحد يخرج هذه المحتويات على البوابة 00 ، وإذا كانت هذه البّت صفر يخرج محتوياتها على البوابة 01 .
28. اكتب برنامج يقرأ بوابة الإدخال رقم 00 إلى ملانهائية ويختبر البيانات التي يقرأها ، فإن كانت فردية يخرجها على البوابة 00 ، وإن كانت زوجية يخرجها على البوابة 01 . احسب أكبر معدل لدخول البيانات لكي يعمل هذا النّظام في الزمن المباشر real time .

الفصل الخامس

برمجة المعالج Z 80

*Programming The Z80
Microprocessor*

٥-١ مقدمة

يعتبر المعالج Z80 صورة متطرفة ومنقحة للمعالج Intel 8085 حيث أن جميع أوامر الشريحة 8085 تتوافق مع الشريحة Z80 ولكن الشريحة Z80 تمتع ببعض المميزات الأخرى الغير موجودة في الشريحة 8085 والتي سنراها في أثناء دراستنا لهذا الجزء . سنتبع في هذا الفصل نفس الطريقة التي اتبناها في الفصل السابق حيث سنقسم أوامر الشريحة Z80 إلى مجموعات وسندرس بالتفصيل من كل مجموعة بعض الأوامر الكثيرة الاستخدام على أننا سنعرض في نهاية الفصل لجدول مختلف لجميع الأوامر حيث يمكن للقارئ الرجوع إليها . وقد تعمدنا أن نتبع نفس طريقة الشرح في الفصل السابق حتى يتمكن من ي يريد أن يتبع المعالج Z80 فقط دون اللجوء إلى مراجعة أي معالج آخر قد لا يحتاج إليه في أثناء تدريبه ، كما أننا سنعرض مقارنات بسيطة بين البروسيسور Z80 و البروسيسور 8085 في بعض المواقف التي تتطلب ذلك حتى تكتمل الفائدة لمن يريد ذلك .

٥-٢ مجموعة أوامر الانتقال

Transfer instructions

يقوم أي أمر من أوامر هذه المجموعة بنقل معلومة (بait) من مكان لأخر حيث المكان الذي تخرج منه المعلومة سنيمه المصدر Source وسترمز له أحياناً بالرمز sss وهذا المكان قد يكون مسجلاً داخل شريحة المعالج وقد يكون بaitاً من بaitات الذاكرة ، وأما المكان الذي ستذهب إليه المعلومة فسوف نسميه الهدف Destination وسترمز له أحياناً بالرمز ddd وهذا المكان أيضاً قد يكون مسجلاً داخل شريحة المعالج وقد يكون بaitاً من بaitات الذاكرة كما سنرى . تمتع جميع أوامر الانتقال الخاصة بالشريحة Z80 بأن لها نفس الأحرف مهما كان مصدر أو هدف المعلومة وهذه الأحرف هي LD التي هي اختصاراً لكلمة LOAD أو حمل ، وذلك على العكس من الشريحة 8085 التي تستخدم عدداً أكثر من الأحرف والتي منها MOV و MVI و LDA وغير ذلك من الصور التي تستخدم كل منها في حالته الخاصة كما رأينا في الفصل السابق ، لذلك سنتناول أوامر الانتقال في حالة الشريحة Z80 على حسب مصدر وهدف المعلومة كما يلى :

٥-٢-١ نقل معلومة من مسجل إلى مسجل آخر

الصورة العامة لهذا الأمر هي :

LD ddd,sss

ddd ← مسجل sss

و معناه تحميل المسجل ddd بمحطيات المسجل sss ، لاحظ أن الذى يتم نقله من المسجل sss هو صورة من المحتويات فقط أما محتويات المسجل فتظل كما هى لا تتغير . من الأمثلة على ذلك ما يلى :

- الأمر LD A,B حيث سيقوم هذا الأمر بنقل (نسخ) محتويات المسجل B (المصدر) إلى المسجل A (الهدف) .

• الأمر LD C,H سيقوم بتحميل المسجل C بمحطيات المسجل H .

• الأمر LD B,B سيقوم بتحميل المسجل B بمحطيات المسجل B . لاحظ أن تأثير هذا الأمر يكفى تماماً "لا تعمل شيئاً" وهذه العملية تكون مهمة جداً فى الكثير من التطبيقات ولذلك فقد أفرد لها أمر خاص بها وهو الأمر NOP أوى Operation أو لا تعمل شيئاً ، وهذا الأمر سرarah فى الكثير من التطبيقات القادمة إن شاء الله . الشفرات الستعشرية لجميع أوامر الانتقال بين جميع المسجلات بعضها البعض يوضحها شكل (5-1) . بالنظر لهذا الشكل نجد أنه إذا أردنا مثلاً نقل محتويات المسجل A إلى المسجل L نستخدم الأمر LD L,A الذي شفرته من شكل (1-5) هي 6F . جميع أوامر نقل المعلومة من مسجل إلى آخر تكون من بait واحد فقط تسمى OP Code وهى اختصار Operation Code أو شفرة العملية .

5-2-2 تحميل مسجل بمعلومة فورية أو ثابت Constant

في الكثير من التطبيقات تحتاج لتحميل مسجل من المسجلات بثابت معين ، في هذه الحالة تكون الصورة العامة لمثل هذه الأوامر كما يلى :

LD ddd,data8

ddd ← data8

و معناه حمل المسجل ddd بالمعلومة الفورية أو الثابت data8 ، لاحظ أن data8 يقصد بها ثابت مكون من ثمانية بتات ، جميع أوامر هذه المجموعة تتكون من اثنين بait واحد هى شفرة الأمر Op Code والثانية هى البايت data8 أو الثابت . شكل (2-5) يبين الشفرات الستعشرية لعملية تحميل أي مسجل من المسجلات بثابت data8 . من هذا الشكل نلاحظ أنه لتحميل المسجل D بالقيمة 55H مثلاً نستخدم الأمر LD D,55H والذي ستكون شفرته الستعشرية كما يلى :

16

55H

حيث 16 (البait الأولى) هي op code كما في شكل (5-2) أما البait الثانية فهي قيمة الثابت المراد تحميله في المسجل D وهو 55H ، لاحظ أن H بعد الرقم تعني أن هذا الرقم ممثل في النظام المستعشرى .

مسجل الهدف Destination register	مسجل المصدر Source register						
	A	B	C	D	E	H	L
A,	7F	78	79	7A	7B	7C	7D
B,	47	40	41	42	43	44	45
C,	4F	48	49	4A	4B	4C	4D
D,	57	50	51	52	53	54	55
E,	5F	58	59	5A	5B	5C	5D
H,	67	60	61	62	63	64	65
L,	6F	68	69	6A	6B	6C	6D

شكل (5-1) الشفرة المستعشرية لجميع أوامر الإنتقال بين المسجلات

A	B	C	D	E	H	L
3E	06	0E	16	1E	26	2E
data8						

شكل (5-2) الشفرات المستعشرية لأوامر تحميل المسجلات بمعلومة فورية أو ثابت data8

مثال 1.5

المطلوب تحميل المسجلات H, E, D, C, B, A بالمعلومات الفورية أو الثوابت الآتية : 01, 02, 03, 04, 05, 06 وبعد ذلك يتم إجراء إزاحة دورانية لهذه المحتويات بحيث تذهب محتويات المسجل A إلى المسجل B وتذهب محتويات المسجل B إلى المسجل C وهكذا إلى أن تذهب محتويات المسجل H إلى المسجل A . شكل (4-2) في الفصل السابق يبين رسمًا توضيحيًا ومخطط السيير لهذا البرنامج ، أما البرنامج بلغة الأسsembli والشفرات المستعشرية فإنه موضح في شكل (5-3) . شكل (5-3) في الفصل السابق يبين نفس هذا البرنامج مكتوبًا بلغة الأسsembli الخاصة بالشريحة 8085 وبمقارنة الشفرات المستعشرية لكلا البرنامجين في الشكلين (5-3 و 4-3) نجد أن هناك تطابقاً تاماً في الشفرات المستعشرية في الحالتين وهذا يبين مدى التطابق بين الشريحتين Z80 و 8085 فإن أي برنامج مكتوبًا بالشفرات المستعشرية للشريحة 8085 يمكن تنفيذه باستخدام الشريحة Z80

وأما البرامج المكتوبة بالشفرات السمعية للشريحة Z80 فليس بالضرورة أنه يمكن تنفيذه على الشريحة 8085 وذلك لأن الشريحة Z80 تحتوى على عدد أكثر من الأوامر الغير معرفة لدى الشريحة 8085 .

العناوين	شفرات أسمبلي	شفرات سمعية
E000	LD A,01	3E 01
E002	LD B,02 ,	06 02
E004	LD C,03	0E 03
E006	LD D,04	16 04
E008	LD E,05	1E 05
E00A	LD H,06	26 06
E00C	LD L,A	6F
E00D	LD A,H	7C
E00E	LD H,E	63
E00F	LD E,D	5A
E010	LD D,C	51
E011	LD C,B	48
E012	LD B,L	45

شكل (3-5) برنامج الأزاحة الدورانية

لتتنفيذ البرنامج الموجود في شكل (3-3) فإنه يمكن أن ندخل في الذاكرة RAM ونكتب الشفرات السمعية ابتداء من العنوان E000 وبعد الانتهاء من كتابة البرنامج في الذاكرة ننفذه باستخدام الأمر GO E000 . أما إذا كان الميكروكمبيوتر الذي نستخدمه به الأسمبلر الخاص بالمعالج Z80 فإنه يمكننا في هذه الحالة كتابة البرنامج بلغة الأسمبلي مباشرة ثم تنفيذه وسنترك تفاصيل عملية إدخال البرنامج لأنها تختلف من شخص لآخر ولكننا ننصح أن تكتب البرنامج الأولى في مرحلة التدريب بالشفرات السمعية ثم بعد ذلك تكتب بلغة الأسمبلي وننصح أيضاً أن يتم تنفيذ البرامج الأولى بطريقة الخطوة خطوة حتى يتمكن

المتدرب من ملاحظة تأثير كل أمر على حده ومتابعة تحميل المسجلات وانتقال البيانات من مسجل لأخر .

يمكن أيضا تحميل زوج من المسجلات بمعلومة مكونة من 16 بتا كما يلى :

LD rp,data16

rp زوج المسجلات data16

حيث rp ترمز لأى زوج من أزواج المسجلات المتاحة فى المعالج Z80 وهى IY, IX, SP, HL, DE, BC (راجع شكل (2-6) لترى أزواج المسجلات المتاحة فى الشريحة Z80) . شكل (4-5) يبين الشفرات الستعشرية المصاحبة لكل زوج فى هذا الأمر . لاحظ أن جميع هذه الأوامر ستكون من ثلاثة بآيات ، واحدة (الأولى) ستكون شفرة الأمر op code والثانى التاليتان ستحتويان المعلومة data16 المكونة من 16 بت كما ذكرنا فيما عدا حالة الزوج IX و IY فإن الأمر سيتكون من أربعة بآيات ، إثنان لشفرة الأمر op code وإثنان للمعلومة data16 . لاحظ أننا مجازا فقط فى هذا المكان نطلق على المسجلات IX, SP, IY, IX كلمة أزواج ولكن ليكن راسخا فى العلم أن كل واحدة منها عبارة عن مسجل واحد مكون من 16 بت ولا يمكن تقسيمه إلى مسجلين كما هو الحال فى الزوج BC مثلما الذى يمكن استخدامه كمسجلين B و C . كمثال على ذلك فإن الأمر :

LD HL,E100

سيحمل الزوج HL بالمعلومة E100H حيث ستذهب البآيت الأولى من المعلومة (E0) إلى المسجل L والبآيت الثانية من المعلومة (E1) ستذهب إلى المسجل H . الشفرات الستعشرية لهذا الأمر ستكون كالتالى :

21

00

E1

5-2-3 نقل معلومة من مسجل إلى الذاكرة والعكس

لنقل معلومة من مسجل في داخل المعالج Z80 إلى أي مكان في الذاكرة أو العكس يمكن استخدام طرق من ثلاثة طرق متاحة لهذا الغرض وهي كالتالى :

BC	DE	HL	SP	IX	IY
01 data16	11 data16	21 data16	31 data16	DD21 data16	FD21 data16

شكل (4-5) الشفرات الستعشرية لأوامر تحميل أزواج المسجلات بمعلومة فورية أو ثابت data16

٥-٣-١ الطريقة المباشرة Direct addressing

في هذه الطريقة يكون عنوان البايت المراد التعامل معها موجوداً في الأمر نفسه (في البايت الثانية والثالثة) ولذلك فإن مثل هذه الأوامر تكون دائماً من ثلاثة بaitات واحدة هي شفرة الأمر op code واثنتان للعنوان المراد التعامل معه . هناك أمران فقط تحت هذه المجموعة ، أمر خاص بنقل المعلومات من مسجل التراكم إلى الذاكرة والأخر خاص بنقل المعلومات من الذاكرة إلى مسجل التراكم A ولذلك فانتا نلاحظ أن التعامل بالطريقة المباشرة لا يكون إلا بين مسجل التراكم فقط والذاكرة ، فإذا أردنا نقل معلومة مثلاً من المسجل B إلى الذاكرة بهذه الطريقة فانتا ننقل المعلومة أولاً إلى المسجل A ثم منه إلى الذاكرة . الأمر الأول في هذه المجموعة هو الأمر :

LD A,(addr)
A ← (addr) المسجل

الشفرة السعشرية لهذا الأمر هي :

3A

البايت ذات القيمة الصغرى من العنوان
addr
البايت ذات القيمة العظمى من العنوان
addr

حيث سيقوم هذا الأمر بنقل محتويات بايت الذاكرة التي عنوانها في الاثنين باليت الثانية والثالثة في الأمر نفسه إلى مسجل التراكم A . ونؤكد هنا على كلمة محتويات حتى لا يظن البعض أن العنوان نفسه هو الذي يوضع في المسجل A كما يوحي شكل الأمر لأول وهلة ولقد تم وضع قوسين حول كلمة addr في الصورة الحرافية للأمر للتأكيد على ذلك ولأنه بدون هذين القوسين لن يستطيع الأسمبلر التمييز بين ما إذا كان الرقم addr عنواناً أم معلومة فوريّة مطلوب تحميلها في المسجل A . كمثال على ذلك انظر إلى الأمر :

LD A,(E100)

حيث يقوم هذا الأمر بتحميل المسجل A بمحتويات العنوان E100 من الذاكرة .
الأمر الثاني من أوامر هذه المجموعة هو الأمر :

LD (addr),A
(addr) ← A المسجل

والشفرة السعشرية لهذا الأمر هي :

32

البايت ذات القيمة الصغرى من العنوان
addr
البايت ذات القيمة العظمى من العنوان
addr

حيث سيقوم هذا الأمر بنقل محتويات المسجل A إلى بايت الذاكرة التي عنوانها موجود في البايت الثانية والثالثة من الأمر نفسه ، أي أن هذا الأمر يقوم بالعملية العكسية للأمر السابق . كمثال على ذلك انظر إلى الأمر :

32

00

E1

حيث سيقوم هذا الأمر ب تخزين محتويات المسجل A في بait الذاكرة التي عنوانها E100 . بذلك تكون قد انتهينا من الطريقة المباشرة لعنونة أو التعامل مع الذاكرة . إن هذه الطريقة كما رأينا مثلها مثل أن تقول لصديقك أحمد يأخذ يأخذ محمد وأنت مسافر إلى الرياض أرجوك أن تعطى هذا الخطاب لأخي محمد هناك وعنوانه موجود على الخطاب مباشرة ، هنا صديقك أحمد الذي يمثل المعالج الذي سينفذ الأمر سيعرف عنوان أخيك في الرياض من على الخطاب مباشرة حيث الخطاب في هذه الحالة يعتبر الأمر المطلوب تفيذه .

5-2-3 الطريقة غير المباشرة Indirect addressing

في هذه الطريقة لا يكون عنوان الباريت المراد التعامل معها في الذاكرة موجوداً مباشرة في الأمر نفسه ولكنه يكون موجوداً في مكان آخر وعلى المعالج الذهاب إلى هذا المكان أولاً وقبل تفاصيل الأمر لمعرفة العنوان . هذا المكان يكون زوجاً من المسجلات ولا بد أن يكون زوجاً لأن العنوان كما نعرف دائماً يتكون من 16 بت ، وعادة يكون هذا الزوج هو الزوج HL . الصورة العامة للأمر في هذه الحالة بلغة الأسمبلية تكون كما يلى :

LD ddd,(HL) ←
ddd المسجل (HL)

حيث سيقوم هذا الأمر بنقل محتويات بait الذاكرة التي عنوانها في زوج المسجلات HL إلى المسجل ddd . الصورة العكسية لهذا الأمر هي :

LD (HL),sss
(HL) sss المسجل

حيث سيقوم هذا الأمر بنقل محتويات مسجل المصدر إلى بait الذاكرة التي يوجد عنوانها في زوج المسجلات HL . شكل (5-5) يبين الشفرات المستعشرية لهذه الأوامر في حالة استعماله مع جميع المسجلات الموجودة في الشريحة Z80 وكما ذكرنا فإن العنوان لابد وأن يكون موجوداً في الزوج HL . هذه الأوامر (أوامر الطريقة غير المباشرة) ستكون جميعها مكونة من بait واحد فقط وهي شفرة الأمر op code . كمثال على ذلك الأمر LD B,(HL) وللذى ستكون شفرته المستعشرية 46 حيث سيقوم هذا الأمر بنقل محتويات الباريت التي عنوانها في الزوج HL إلى المسجل B . لاحظ وجود القوسين حول الزوج HL في جميع

هذه الأوامر للدلالة على أن المقصود هو محتويات العنوان الموجود في HL وليس القيمة الموجودة في HL مباشرة .

	A	B	C	D	E	H	L
ddd ← (HL)	7E	46	4E	56	5E	66	6E
(HL) ← sss	77	70	71	72	73	74	75

شكل (5-5) الشفرات المستعشرية لأوامر الانتقال بين المسجلات والذاكرة بالطريقة غير المباشرة

هناك ميزة يمتاز بها المسجل A عن باقى مسجلات الشريحة فى حالة التعامل بينه وبين الذاكرة بالطريقة غير المباشرة وهى أن عنوان البايت المراد التعامل معها فى هذه الحالة يمكن وضعه فى أى زوج من أزواج المسجلات الأخرى وليس بالضرورة الزوج HL كما سبق ، وذلك كما قلنا كحالة خاصة فقط للمسجل A . شكل (5-6) يبين الشفرة المستعشرية والأسمبلى وما يقوم به كل واحد من هذه الأوامر .

الشفرة الأسمبلى	الشفرة المستعشرية	ما يفعله الأمر
LD A,(BC)	0A	A ← (BC)
LD A,(DE)	1A	A ← (DE)
LD (BC),A	02	(BC) ← A
LD (DE),A	12	(DE) ← A

شكل (5-6) أوامر الانتقال بين المسجل A والذاكرة بالطريقة غير المباشرة

كما رأينا فإن الطريقة غير المباشرة فى التعامل مع الذاكرة مثلاً مثل أن نقول لصديقك أحمد يا أخي يا أحمد وأنت مسافر إلى الرياض خذ هذا الخطاب وأعطه أخي محمد ولكن أرجوك قبل سفرك أن تمر على والدى لنعرف منه عنوان أخي محمد في الرياض لأنني لا أعرفه . هنا صديقك أحمد يمثل المعالج الذى سيقوم بالتنفيذ والوالد يمثل المسجلين HL حيث يحتويان العنوان المراد التعامل معه وللذان لابد من المرور عليهما قبل تنفيذ الأمر . السؤال الآن كيف نضع عنوان ما في زوج من المسجلات ؟ إن هذه العملية بسيطة جداً حيث يستخدم فيها أمر تحويل زوج مسجلات بمعلومة فورية أو ثابت مكون من 16 بت والتي درسناها في الجزء السابق (5-2 تحويل مسجل بمعلومة فورية) .

3-3-3 طريقة الفهرسة لعنونة الذاكرة Indexed addressing

هناك مسجلان لم نتكلم عنهما تفصيلياً إلى الآن وهما المسجلان IX و IY وكل منهما يتكون من 16 بت . هذان المسجلان يختلفان عن أزواج المسجلات الأخرى في أنه لا يمكن استخدام أي منها كمسجلين 8 بتات منفصلين ولكن كل منها يستخدم كمسجل 16 بت مثله في ذلك مثل مسجل عدد البرنامج program أو مؤشر المكدسة stack pointer . هذان المسجلان يستخدمان في عملية الإشارة إلى بaitات الذاكرة تماماً مثل الزوج HL ولكن بإمكانيات أكثر ، ولكل نفهم ذلك انظر إلى الأمر التالي :

LD B,(IX+5)

هذا الأمر سيقوم بتحميل المسجل B بمحتويات بait الذاكرة التي عنوانها عبارة عن حاصل جمع محتويات المسجل IX زائد خمسة . لكي نوضح هذا الأمر افترض الوضع التالي :

B	IX	E105
05	E100	66

بعد تنفيذ الأمر LD B,(IX+5) سيصبح الوضع كالتالي :

B	IX	E105
66	E100	66

الرقم الذي يضاف على محتويات المسجل IX أو المسجل IY يسمى "الإزاحة" displacement وهذه الإزاحة تشغل بait كاملة ولذلك فإن قيمتها ستتراوح بين +127 و -128 . حيث الإزاحة الموجبة معناها إضافة إلى محتويات مسجل الفهرسة (IX, IY) وأما الإزاحة السالبة فمعناها طرح من مسجل الفهرسة . الشفرات العشرية لجميع هذه الأوامر موجودة في جداول الأوامر الموضحة في نهاية هذا الفصل .

3-3 تمارين

استخدم قائمة أوامر الانتقال الخاصة بالشريحة Z80 لحل التمارين الموجودة في الجزء 3-3 في الفصل السابق .

4-5 مجموعة أوامر الحساب Arithmatic Instructions

سندرس في هذا الجزء بعض الأوامر التي تقوم بإجراء العمليات الحسابية الأولية وهي الجمع والطرح ، وكما علمنا من قبل فإن مسجل التراكم لا بد وأن يكون طرفا في أي عملية من هذه العمليات كما أن نتيجة هذه العملية سواء كانت جمعاً أو طرحًا تكون دائماً موجودة في مسجل التراكم A . هناك أيضاً خاصية مهمة في مجموعة أوامر الحساب (ومثلها أيضًا أوامر المتنطق كما سنرى) وهي أنه نتيجة تنفيذ أي أمر من هذه الأوامر فإن الأعلام الموجودة في مسجل الحالة Status Register تتأثر بهذه النتيجة . راجع مسجل الحالة ومحطوياته ومتى يكون أي علم من أعلامه يساوى صفرًا أو واحدًا وذلك في الفصل الثاني . كما ذكرنا فإن طرفًا من طرفى العملية الحسابية أو معامل من معاملاتها لا بد وأن يكون موضوعاً في المسجل A وأما الطرف الثاني أو المعامل الثاني فإن مصدره سيكون واحدًا من أربعة أماكن موضحة كالتالي :

<u>مصدر المعامل الثاني للعملية الحسابية</u>	<u>الرمز المستخدم</u>
مسجل 8 بت من مسجلات المعالج .	reg
بأيت من بآيات الذاكرة عنوانها في HL .	(HL)
ثابت أو معلومة فورية مكونة من 8 بت .	data8
بأيت من بآيات الذاكرة معنونة بطريقة الفهرسة باستخدام المسجلين IX أو IY	(IX+d) (IY+d)

1-4-5 الأمران ADD و SUB

الصورة العامة لأمر الجمع ADD هي :

ADD A,reg
 $A \leftarrow A + reg$

حيث سيقوم هذا الأمر بجمع محتويات المسجل reg مع محتويات المسجل A ووضع النتيجة في المسجل A مع التأثير على الأعلام لاحظ أننا في حالة

أسمبلر الشريحة 8085 كنا نكتب هذا الأمر ADD reg بدون ذكر المسجل A على أساس أنه بديهي أن المعامل الثاني للعملية يكون في المسجل A ، أما في أسمبلر الشريحة Z80 فلابد من ذكر طرف العملية الحسابية بالرغم من أن أحدهما يكون دائما المسجل A وإن كانت بعض المراجع تهمل ذلك . الصور العامة الأخرى لأمر الجمع ستكون كالتالى :

ADD A,(HL)

حيث سيقوم هذا الأمر بجمع محتويات بايت الذاكرة التي يوجد عنوانها في الزوج HL مع محتويات المسجل A وتوضع النتيجة في المسجل A .

ADD A,data8

حيث سيجمع الثابت data8 مع المسجل A وتوضع النتيجة في المسجل A .

ADD A,(IX+d)

ADD A,(IY+d)

حيث سيجمع محتويات المسجل A مع محتويات بايت الذاكرة التي عنوانها عبارة عن محتويات المسجل IX أو IY زائد الإزاحة d وتوضع النتيجة في المسجل A . بنفس الطريقة يمكن كتابة الصورة العامة لأمر الطرح تبعا لمصدر المعامل الثاني كما يلى :

SUB A,reg

A ← A - reg

SUB A,(HL)

A ← A - (HL)

SUB A,data8

A ← A - data8

SUB A,(IX+d)

SUB A,(IY+d)

A ← A - (IX+d)

A ← A - (IY+d)

فى جميع هذه الأوامر يتم طرح المعامل الثاني (reg , (HL) , data8 , (IX+d) , (IY+d)) من المسجل A وتوضع النتيجة في المسجل A ، أى أننا نؤكّد هنا على أن المطروح منه يكون دائما المسجل A . لاحظ أن المسجل A يذكر مع هذا الأمر أيضا وذلك بالطبع لا دخل للمستخدم فيه ولكنه من الشروط التي يفرضها الأسمبلر (كما فى بعض المراجع) ، سنهمل ذكر الشفرات الستعشورية للأوامر ابتداء من هذا الموضع ومن يريد التعرف عليها أو استخدامها فعليه الاستعانة بالأشكال الخاصة بالأوامر فى نهاية هذا الفصل .

مثال 5-2

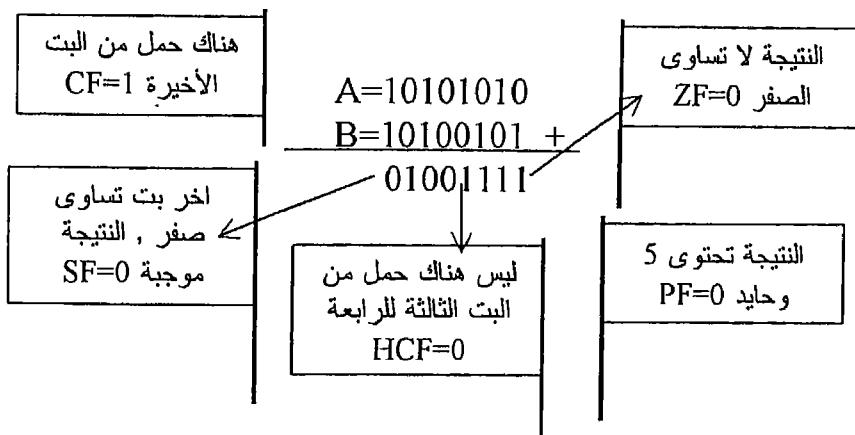
افتراض المحتويات الآتية للمسجلين A و B قبل تنفيذ الأمرين ADD و SUB :



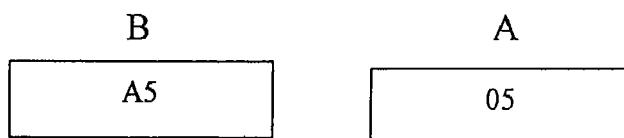
بعد تنفيذ الأمر ADD A,B ستصبح المحتويات كالتالى :



ولكي نرى كيفية تأثر الأعلام بنتيجة هذه العملية سنجرى عملية الجمع على الشفرات الثنائية لكل من الرقمين كما يلى :



الآن افترض أننا نفذنا أمر الطرح SUB B على المحتويات الأولى للمسجلين أى $A=A5H$ و $B=AAH$ فإنه بعد تنفيذ هذا الأمر ستصبح محتويات المسجلين كالتالى:



ولكي نرى كيف تمت عملية الطرح وكيف تأثرت الأعلام سنجري عملية الطرح على الشفرات الثنائية لمحتويات المسجلين A و B كالتالى :

كما نعلم فإن عملية الطرح الثنائى يتم تحويلها إلى عملية جمع حيث سنجمع محتويات المسجل A (المطروح منه) مع المتمم الثنائى لمحتويات المسجل B (المطروح) (انظر الملحق الأول فى نهاية الكتاب لمراجعة عمليات الجمع والطرح الثنائى) . المتمم الثنائى لمحتويات المسجل B (10100101) هو 01011011 وبذلك تصبح عملية الطرح عملية جمع كالتالى :

$$\begin{array}{r}
 A = 10101010 \\
 B = 01011011 = \text{المتم الثنائي لمحتويات المسجل} \\
 \hline
 & 00000101
 \end{array}$$

وبناء على ذلك ستكون الأعلام كالتالى :

- طالما أن البت الأولى لا تساوى صفرًا فالنتيجة لا تساوى صفرًا ويكون علم الصفر $ZF=0$.
- تحتوى النتيجة على عدد زوجي من الوحدات (اثنين) لذلك سيكون علم الباريتى واحدا ، $PF=1$.
- هناك حمل من البت الثالثة إلى البت الرابعة في حالة الجمع لذلك فعلم الحمل النصفي $HCF=0$.
- آخر بت (رقم 7) تساوى صفرًا ، لذلك فالنتيجة موجبة وعلم الإشارة يكون دائما مساويا لمحتويات آخر بت ، إذن $SF=0$.
- المفروض في عمليات الطرح يهمنا أن نعرف إذا كان هناك استلاف أم لا لأنه في عملية الطرح لن يكون هناك حمل بما أن عملية الطرح قد حولت إلى عملية جمع لذلك فإنه إذا كان هناك حمل في عملية الجمع فإن ذلك يعني أنه لن يكون هناك استلاف في عملية الطرح وسيكون علم الحمل $CYF=0$ وهي الحالة التي معنا الآن والعكس صحيح إذا لم يكن هناك حمل في عملية الجمع . وهذا هو ما طبقناه في حالة العلم HCF

5-4-2 الأمران ADC و SBC

بالنسبة للأمر ADC فإنه يجمع المعامل الثاني سواء كان في مسجل أو ذاكرة أو قيمة فورية مع محتويات المسجل A مع محتويات علم الحمل CY (صفر أو واحد) ويوضع النتيجة في المسجل A . الصورة العامة لهذه الأوامر وعلى حسب مصدر المعامل الثاني ستكون كما يلى :

ADC A,reg

$A \leftarrow A + CY + reg$

ADC A,(HL)

$A \leftarrow A + CY + (HL)$

ADC A,data8

$A \leftarrow A + CY + data8$

ADC A,(IX+d)

$A \leftarrow A + CY + (IX+d)$

ADC A,(IY+d)

$A \leftarrow A + CY + (IY+d)$

يمكننا الآن تكرار نفس القول بالنسبة لأمر الطرح SBC حيث يقوم هذا الأمر بطرح المعامل الثاني سواء كان في مسجل أو ذاكرة أو قيمة فورية مع محتويات

علم الحمل CY (صفر أو واحد) من المسجل A ثم توضع نتيجة الطرح في المسجل A ، نؤكد هنا على أن المطروح منه دائمًا هو المسجل A . الصورة العامة لهذه الأوامر وعلى حسب مصدر المعامل الثاني ستكون كما يلى :

SBC A,reg
 $A \leftarrow A - CY - reg$
SBC A,(HL)
 $A \leftarrow A - CY - (HL)$
SBC A,data8
 $A \leftarrow A - CY - data8$
SBC A,(IX+d)
 $A \leftarrow A - CY - (IX+d)$
SBC A,(IY+d)
 $A \leftarrow A - CY - (IY+d)$

مثال 3-5

E000 E001	LD C,F9H
E002 E003	LD B,23H
E004 E005	LD E,35H
E006 E007	LD D,9AH
E008	LD A,C
E009	ADD A,E
E00A E00B E00C	LD (E100),A
E00D	LD A,B
E00E	ADC A,D
E00F E010 E011	LD (E101),A
E012 E013	LD A,00
E014	ADC A,A
E015 E016 E017	LD (E102),A

شكل (7-5) برنامج المثال 5

المطلوب جمع الرقمين 23F9H و 9A35H ووضع نتيجة الجمع في أماكن الذاكرة E100 و E101 و E102 .

هذا المثال هو المثال رقم 4-11 وقد سبق حله كتطبيق على أمر الجمع مع الحمل في حالة الشريحة 8085 ويبين شكل (4-6) رسمًا توضيحيًا ومخطط السيير والبرنامج لطريقة حل هذا المثال ، لذلك يمكن مراجعته أولاً وسنعيد كتابة البرنامج فقط بلغة الأسsembli الخاصة بالشريحة Z80 في شكل (7-7) . قبل أن نترك أوامر الجمع والطرح يجب أن نفهم جيداً متى يكون من الضروري

استخدام الأمر ADC ؟ ومتى يكون من الضروري عدم استخدامه؟ مثلاً في المثال السابق كان من الضروري عدم استخدام الأمر ADC في عملية الجمع الأولى ($E + C$) ولكن في هذه الحالة لابد من استخدام الأمر ADD خوفاً من أن يكون علم الحمل CY به واحد من أي عملية سابقة ونحن لا ندرى فيجمع مع عملية الجمع الأولى إذا استخدمنا الأمر ADC وتكون النتيجة خطأة . أما في عملية الجمع الثانية ($D + B + CY$) فإنه لابد من استخدام الأمر ADC لأننا نريد هنا أن نأخذ قيمة علم الحمل في الاعتبار .

3-4-5 الأمران INC و DEC

هذان الأمران يستخدمان لزيادة أو إنفاص واحد على أو من محتويات مسجل أو بait من بaitات الذاكرة . الصورة العامة للأمر INC وعلى حسب مكان المعلومة يمكن كتابتها كالتالى :

INC reg

$reg \leftarrow reg + 1$

INC (HL)

$(HL) \leftarrow (HL) + 1$

INC (IX+d)

$(IX+d) \leftarrow (IX+d) + 1$

INC (IY+d)

$(IY+D) \leftarrow (IY+d) + 1$

بنفس الطريقة يمكن كتابة الصورة العامة للأمر DEC كما يلى :

DEC reg

$reg \leftarrow reg - 1$

DEC (HL)

$(HL) \leftarrow (HL) - 1$

DEC (IX+d)

$(IX+D) \leftarrow (IX+d) - 1$

DEC (IY+d)

$(IY+d) \leftarrow (IY+d) - 1$

4-4-5 العمليات الحسابية على أزواج المسجلات

عند إجراء العمليات الحسابية على أزواج المسجلات يلعب الزوج HL دور مسجل التراكم من حيث أن المعامل الأول في العملية الحسابية لابد وأن يكون في الزوج HL ونتيجة العملية الحسابية تذهب دائماً إلى الزوج HL . يجب أن نعلم أنه عند إجراء العمليات الحسابية على أزواج المسجلات أن الأعلام لا تتأثر

بهذه العمليات فى الكثير من الأحيان ويجب النظر فى حالة كل أمر منفصلة .
الصورة العامة للأمرتين INC و DEC فى هذه الحالة هي :

INC rp

$rp \leftarrow rp + 1$

DEC rp

$rp \leftarrow rp - 1$

حيث rp ترمز لأى زوج من أزواج المسجلات SP, HL, DE, BC أو مسجل من المسجلات الـ 16 بت وهى المسجل IX أو المسجل IY . كامثلة على ذلك انظر إلى الأوامر التالية :

INC HL

DEC SP

INC IX

الأمر الأول سيزيد واحدا على محتويات المسجلين HL والثانى سينقص واحدا من محتويات المسجل SP والثالث سيزيد واحدا على محتويات المسجل IX . الأوامر DEC rp و INC rp ليس لها تأثير على الأعلام .
من أوامر الجمع والطرح التى تجرى على أزواج المسجلات ما يلى :

ADD HL, rp

$HL \leftarrow HL + rp$

ADC HL, rp

$HL \leftarrow HL + rp + CY$

SBC HL, rp

$HL \leftarrow HL - rp - CY$

فى جميع هذه الأوامر ترمز rp لأى زوج من الأزواج SP, HL, DE, BC ما عدا المسجلين IX, IY فلا يمكن استخدامهما مع هذه الأوامر . لاحظ أيضا أن أمر الطرح الوحيد المتاح هو أمر الطرح مع الحمل SBC ولذلك فإنه عند إجراء أى عملية طرح بدون أخذ علم الحمل فى الحساب يجب فى هذه الحالة التأكد من أن علم الحمل يساوى صفراء . الأمر ADD HL, rp ليس له تأثير على الأعلام وأما الأوامر ADD HL, rp و ADC HL, rp فيؤثران على الأعلام ما عدا علم الحمل النصفى HC . نؤكد هنا على أن المطروح منه فى الأوامر السابقة هو المسجلين HL . هناك بعض الأوامر التى تسمح للمسجل IX أو المسجل IY بأن يلعب دور مسجل التراكم فى عمليات الجمع على أزواج المسجلات ، وهذه الأوامر هى :

IX \leftarrow IX + BC	ADD IX, BC
-------------------------	------------

IX \leftarrow IX + DE	ADD IX, DE
-------------------------	------------

IX \leftarrow IX + SP	ADD IX, SP
-------------------------	------------

IX \leftarrow IX + IX	ADD IX, IX
-------------------------	------------

IY \leftarrow IY + BC	ADD IY, BC
-------------------------	------------

IY \leftarrow IY + DE	ADD IY, DE
-------------------------	------------

$IY \leftarrow IY + SP$
 $IY \leftarrow IY + IY$

ADD IY,SP
ADD IY,IY

جميع هذه الأوامر تؤثر على الأعلام ما عدا علم الحمل النصفى HC وأيضا جميع هذه الأوامر ليس لها نظير لعملية الطرح .

5-4-5 أمر المقارنة Compare Instruction

هناك أمر واحد فقط للمقارنة حيث أن عملية المقارنة لا تجرى على أي معلومة مكونة من 16 بت . لكي تتم عملية المقارنة فإن أحد المعاملين لابد وأن يكون في المسجل A والمعامل الآخر يكون إما في مسجل من مسجلات المعالج أو في بait من بaitات الذاكرة . عند تنفيذ أمر المقارنة يقوم المعالج بطرح محتويات المعامل الثاني من محتويات المسجل A وتهمل نتيجة الطرح تماما ولا تتغير محتويات المسجل A نتيجة هذه العملية ولكن الذي يتأثر فقط بهذه العملية هو الأعلام . الصورة العامة لأمر المقارنة وعلى حسب مصدر المعامل الثاني يمكن كتابتها كما يلى :

A - reg	CP reg
A - (HL)	CP (HL)
A - data8	CP data8
A - (IX+d)	CP (IX+d)
A - (IY+d)	CP (IY+d)

لاحظ أنتا لم نكتب السهم الذى يوضح أين تذهب نتيجة عملية الطرح على أساس أن النتيجة تهمل كما ذكرنا . كمثال على ذلك انظر إلى الأوامر التالية :

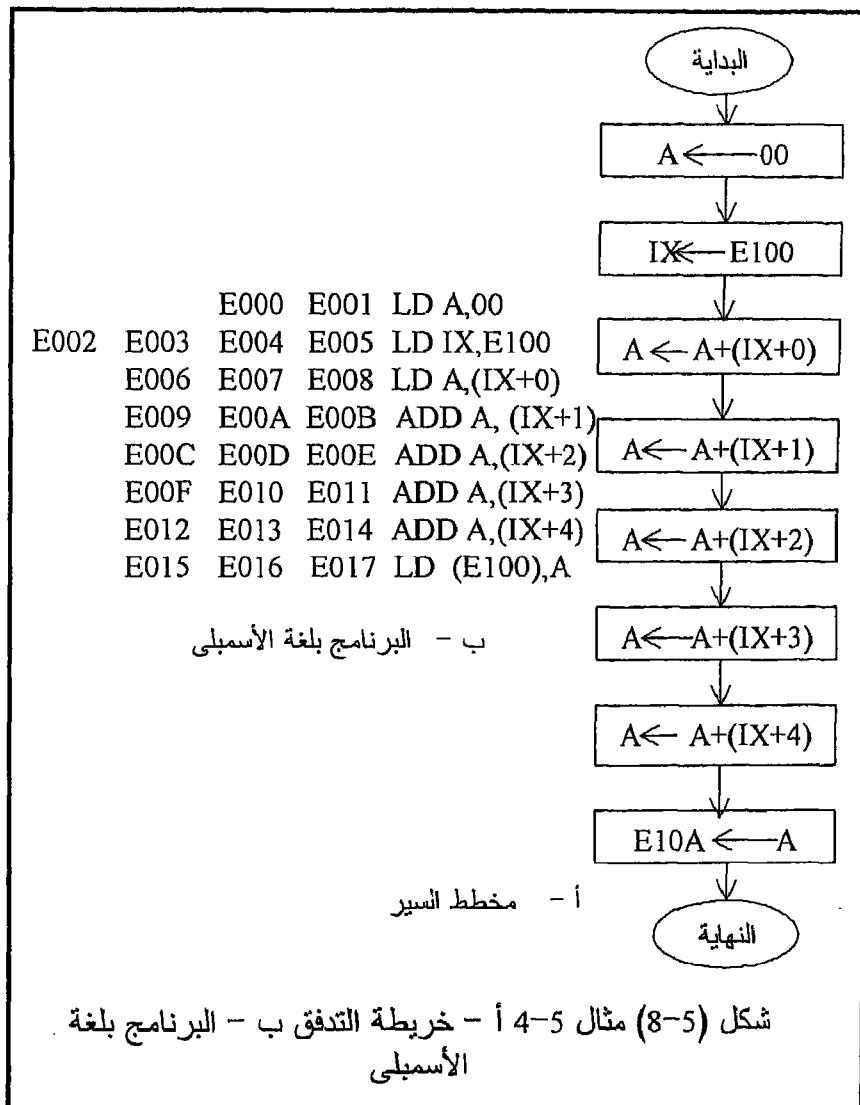
CP B
CP 3FH
CP (IY+9)

حيث سيقارن الأمر الأول محتويات المسجل B مع محتويات المسجل A وسيقارن الأمر الثاني محتويات المسجل A مع الثابت أو القيمة الفورية 3FH وأما الأمر الثالث فسيقارن محتويات المسجل A مع محتويات بait الذاكرة التي عنوانها تسعه زائد محتويات المسجل IY .

مثال 4-5

اكتب برنامجا يجمع محتويات الخمس بaitات E104, E103, E102, E101, E100 ويضع النتيجة في البait E10A على اعتبار أن النتيجة المتوقعة لن تزيد عن بait واحدة ، أى لن يكون هناك حمل على الإطلاق . هذا المثال من الممكن أن يكون تدريبيا جيدا على طرق الاتصال بالذاكرة التي تمت دراستها حتى الآن . شكل (5-8) يبين مخطط السير والبرنامنج لهذا المثال مستخدمين طريقة الفهرسة

مع المسجل IX للاتصال بالذاكرة ، حاول كتابة البرنامج مرة أخرى مستخدما
الطريقة غير المباشرة مع الزوج HL .



5-5 تمارين

حل التمارين الموجودة في الجزء 4-5 في الفصل الرابع مستخدما أوامر لغة الأسمبل الخاصة بالشريحة Z80 .

5-6 مجموعة أوامر القفز Jump Instructions

القاعدة العامة أن المعالج يقوم بتنفيذ البرنامج حسب ترتيب الأوامر الموجودة فيه من أول البرنامج إلى نهايته . ولقد كنا حريصين في جميع الأمثلة السابقة على الحفاظ على هذه القاعدة ، ولكن هناك بعض المواقف أو قل بعض التطبيقات التي تتطلب الخروج على هذه القاعدة لأن يطلب منك مثلاً تنفيذ عملية معينة (عدد من الأوامر) عدد معين أو حتى عدد لا نهائي من المرات . فعندما يكون المعالج مثلاً مراقباً لدرجة الحرارة في عملية صناعية معينة فإن عليه أن يقرأ درجة الحرارة ويقارنها بدرجة حرارة مخزنة في الذاكرة كمراجع وإذا زادت الحرارة عن حد معين يقوم المعالج بضرب جرس إنذار ، وإذا نقصت عن حد معين يشغل سخان لزيادتها ، مثل هذا البرنامج سيكون عبارة عن مجموعة من الأوامر التي تتفق إلى ملائمة طالما أن المعالج يرافق درجة الحرارة . لقد أتاح المعالج هذه العملية بتوفير بعض الأوامر التي تمكنك كمبرمج من التفاز بعملية التنفيذ من مكان آخر خلال البرنامج ، وهناك ثلاثة أنواع من القفز توفرها الشريحة Z80 وهي كما يلى :

5-6-1 القفز غير المشروط Unconditional jump

عند تنفيذ أي عملية قفز غير مشروط ينتقل المعالج بعملية التنفيذ إلى المكان الجديد دون أي قيد أو شرط ، وهناك أمر واحد فقط من أوامر الشريحة Z80 يقوم بهذه العملية والصورة العامة لهذا الأمر هي :

JP addr

عند تنفيذ هذا الأمر يوضع العنوان addr الذي سيتم القفز إليه في عداد البرنامج PC فيصبح الأمر الموجود عند هذا العنوان addr هو الأمر الذي عليه الدور في التنفيذ . لاحظ أن هذا الأمر يتكون من ثلاثة بaites واحدة هي شفرة الأمر واثنتان للعنوان addr الذي سيتم القفز إليه . إن القفز باستخدام الأمر قد يكون إلى الأمام في البرنامج وقد يكون إلى الخلف . إذا كان القفز إلى الأمام سينتج عن ذلك وجود جزء من البرنامج لن ينفذ على الإطلاق وهو الجزء الذي يقع بين أمر القفز JP addr والأمر الذي سيتم القفز إليه . أما إذا كان القفز إلى الخلف فإنه سينتج عن ذلك ما يسمى بالحلقة اللاحائية Infinite loop والتي سيستمر المعالج في تنفيذها إلى ملائمة . شكل (4-8) يبين خريطة تدفق لعملية القفز غير المشروط بنوعيها الأمامي والخلفي .

5-6-2 القفز المشروط Conditional jump

كما يوحى الاسم فإنه في هذا النوع من القفز لن يتم القفز إلا إذا تحقق شرط معين ، أما إذا لم يتحقق هذا الشرط فإن البرنامج يتم تنفيذه في التتابع الطبيعي حيث سينفذ الأمر الذي بعد أمر القفز مباشرة . إن شروط القفز توضع دائمًا على الأعلام التي في مسجل الحالة SR ، فيمكنك مثلاً أن تجعل القفز مشروطاً بـأن تكون النتيجة صفرًا أو تجعله مشروطاً بأن تكون النتيجة سالبة وهكذا . حيث أن هناك خمسة أعلام واحد منها وهو علم الحمل النصفي HC لا يستخدم كشرط في عمليات القفز فإنه يتبقى أربعة أعلام يمكن أن تستخدم في أوامر القفز المشروط كما يلى :

ZF=1	JP Z,addr	اقفز إذا كانت النتيجة صفرًا
ZF=0	JP NZ,addr	اقفز إذا كانت النتيجة ليست صفرًا
SF=1	JP M,addr	اقفز إذا كانت النتيجة سالبة
SF=0	JP P,addr	اقفز إذا كانت النتيجة موجبة
CF=1	JP C,addr	اقفز إذا كان هناك حمل
CF=0	JP NC,addr	اقفز إذا لم يكن هناك حمل
PF=1	JP PE,addr	اقفز إذا كانت الباريتى زوجية
PF=0	JP PO,addr	اقفز إذا كانت الباريتى فردية

لاحظ أن عدد هذه الأوامر ثمانية ، إثنان منها لكل علم من الأعلام الأربع تمثل جميع الحالات التي يمكن أن يكون فيها هذا العلم صفرًا أو واحدًا . أيضاً جميع هذه الأوامر لابد وأن تتكون من ثلاثة بaites واحدة هي شفرة الأمر op code واثنتان للعنوان الذي سيتم القفز إليه . إن النتيجة التي سيتوقف عليها أمر القفز هي آخر نتيجة تأثرت بها الأعلام ، ولذلك فإنه قبل أن نكتب أي أمر من أوامر القفز المشروط يجب أن ندرس جيداً هل الأمر السابق لأمر القفز يؤثر على الأعلام أم لا .

5-6-3 القفز النسبي Relative jump

هناك أنواع أخرى من القفز متاحة لدى المعالج Z80 مثل القفز النسبي والقفز للبرامج الفرعية والعودة منها وستترك الكلام عن هذه الأنواع حيث سيتم شرحها بالتفصيل في فصول خاصة بذلك .

مثال 5-5

اكتب برنامجاً يقرأ محتويات البایت E100 باستمرار إلى مالانهاية ثم يختبر هذه المحتويات بحيث إذا كانت صفرًا يضع واحداً في المسجل B وإذا كانت سالبة

يضع اثنين في المسجل B وإذا كانت موجبة يضع أربعة في نفس المسجل . شكل (9-4) في الفصل السابق يبين مخطط السير لهذا البرنامج وسنعيد فقط كتابة البرنامج بلغة الأسمبلى الخاصة بالشريحة Z80 كما فى شكل (5-9) .

E000	E001	E002	LD HL,E100
E003	E004		LD A,00
E005	ADD		A,(HL)
E006	E007	E008	JP NZ,E00E
E009	E00A		LD B,01
E00B	E00C	E00D	JP E000
E00E	E00F	E010	JP P,E016
E011	E012		LD B,02
E013	E014	E015	JP E000
E016	E017		LD B,04
E018	E019	E01A	JP E000

شكل (5-9) برنامج المثال 5-5

7-5 مهمة أخرى للأسمبلر

المهمة الوحيدة التي عرفناها للأسمبلر حتى الآن هي مهمة تحويل شفرات الأسمبلر إلى شفرات ثنائية أو لغة ماكينة ، ولكن لحسن الحظ فإن هناك مهام أخرى يستطيع الأسمبلر القيام بها ومن شأن هذه المهام أن تريح المبرمج وتتوفر عليه الكثير من المجهود . جزء 4-7 في الفصل السابق تناول هذه المهام كما تناول أيضا عملية تقسيم أي أمر من أوامر لغة الأسمبلر إلى أجزاء مختلفة وكيف يتعرف الأسمبلر على هذه الأجزاء ، لذلك فإننا لن نكرر هذا الجزء هنا ولكن نحن القارئ لمراجعته في الفصل السابق مع الأخذ في الاعتبار التوارق البسيطة بين شفرات الأسمبلر الخاصة بالشريحة Z80 والشريحة 8085 .

5-8 أوامر الإدخال والإخراج Input Output Instructions

إلى الآنرأينا كيف نبرم杰 شريحة المعالج وكيف نحرك المعلومات داخلها من مسجل إلى مسجل آخر ومن أي مسجل إلى الذاكرة والعكس ، ولكن لم نعرف

حتى الآن كيف نظهر معلومة على شاشة عرض مثلاً أيا كان نوع هذه الشاشة ، أو كيف ندخل معلومة إلى المعالج من خلال لوحة مفاتيح على سبيل المثال . إن لوحة المفاتيح وشاشة العرض يعتبران مثالين من العديد من الأمثلة التي تحتاج إلى عمليات الإخراج والإدخال . حينما يستخدم المعالج للتحكم في أي متغير في عملية صناعية ول يكن مثلاً درجة الحرارة فإنه لا بد من إدخال درجة الحرارة إلى المعالج بعد تهيئتها ووضعها في الصورة المناسبة لذلك ، وكذلك إذا أراد المعالج رفع درجة حرارة العملية الصناعية أو ضرب جرس إنذار فإنه يخرج إشارة معينة على بوابة إخراج تؤخذ وتهيأ في الصورة المناسبة للجهاز الذي ستدبر إليه سواء كان سخاناً أو جرساً . إن جميع عمليات الإدخال والإخراج تتم من خلال ما يسمى ببوابات الإدخال والإخراج والتعامل مع هذه البوابات دائماً ينقسم إلى قسمين : قسم خاص بالبناء الإلكتروني لهذه البوابات وكيفية توصيلها مع المعالج وهذا القسم سندرسه بالتفصيل في فصل قادم إن شاء الله ، والقسم الآخر هو كيفية برمجة المعالج للتعامل مع هذه البوابات وهو موضوع دراستنا في هذا الجزء حيث سندرس الأوامر الخاصة بذلك وسنفترض في دراستنا لهذا الجزء أن القاريء لديه على الأقل بوابة إدخال input port وبوابة إخراج output port موصلين في الميكروكمبيوتر الذي يستخدمه في التدريب وكتابة البرامج .

5-8-1 أوامر الإدخال Input Instructions

الصورة العامة لأمر الإدخال هي :

IN A, no

محطيات البوابة رقم no. \leftarrow المسجل A

حيث IN هي اختصار لكلمة Input بمعنى ادخل ، وسيقوم هذا الأمر بإدخال المعلومة الموجودة على بوابة الإدخال رقم no. إلى مسجل التراكم A . لاحظ أن عملية الإدخال بهذا الأمر تكون دائماً على المسجل A حيث يمكن نقل المعلومة بعد ذلك إلى أي مكان آخر . هذا الأمر يتكون من الثنين بايت ، واحدة هي شفرة الأمر op code والأخرى هي رقم البوابة التي سيتم التعامل معها . ولذلك فإنه طالما أن رقم البوابة يشغل بايت كاملة فإن ذلك يعني أنه يمكن التعامل مع 2^{16} = 256 بوابة إدخال تبدأ من البوابة رقم 00H إلى البوابة رقم FFH . انظر شفرة هذا الأمر في جداول الأوامر في نهاية هذا الفصل . هناك طريقة غير مباشرة للتعامل مع بوابات الإدخال والصورة العامة لها كالتالي :

IN reg,(C)

بوابة الإدخال التي رقمها في المسجل C \leftarrow المسجل reg

حيث سيقوم هذا الأمر بإدخال المعلومة الموجودة في بوابة الإدخال التي رقمها في المسجل C إلى المسجل reg الذي هو أي مسجل من مسجلات المعالج وهذه ميزة عظيمة لم تكن موجودة في المعالج 8085 .

5-8 أوامر الأخرج Output Instruction

الصورة العامة لأمر الأخرج هي :

OUT no ,A

حيث OUT هي اختصار لكلمة Output التي تعنى إخراج ، وسيقوم هذا الأمر بإخراج المعلومة الموجودة في المسجل A إلى بوابة الإخراج التي رقمها no.

هذا الأمر أيضا يشغل اثنين بait واحدa هي شفرة الأمر والأخرى هي رقم البوابة المراد التعامل معها ، ولذلك فإنه بهذا الأمر يمكن التعامل مع $2^8 = 256$ بوابة إخراج تبدأ من البوابة رقم 00H وتنتهي بالبوابة رقم FFH . الطريقة الغير مباشرة لهذا الأمر هي :

OUT (C),reg

حيث سيقوم هذا الأمر بإخراج المعلومة الموجودة في المسجل C

مسجل من مسجلات المعالج إلى بوابة الإخراج التي يوجد رقمها فى المسجل C.

مثال 6-5

افتراض أن لدينا خط إنتاج في أحد المصانع تعبر عليه المنتجات ، وفي أثناء العبور فإن كل منتج يقطع خلية صوتية فتعطى نبضة كهربائية على خرجها . خرج هذه الخلية موصل على البت رقم 0 في بوابة الإدخال رقم 00H والمطلوب هو كتابة برنامج يعد هذه المنتجات ويخرج العدد على بوابة الإخراج رقم 00H . شكل (4-11 و 4-12) في الفصل السابق يبينان رسماً توضيحيًا ومخطط السير لهذا المثال ، أما البرنامج فتمت إعادةه كما في شكل (5-10) .

5-9 مجموعة أوامر المنطق

Logic Instructions

العمليات المنطقية التي يستطيع المعالج Z80 القيام بها هي العمليات XOR, NOT, OR, AND يقوم القارئ بمراجعتها في جداول الأوامر الملحة في آخر الفصل . كما ذكرنا سابقاً فإن العمليات المنطقية مثلها مثل العمليات الحسابية لابد وأن يكون المسجل A طرفاً فيها كما أن النتيجة توضع في المسجل .

المسجل B سيكون عداد المنتج LD B,00;
 المسجل C يحتوى رقم البوابة التى سنخرج عليها LD C,00;
 HERE1: IN A,00 ; القراءة بوابة الإدخال إلى المسجل A
 مقارنة الإشارة بصفر CP 00 ;
 طالما أن الإشارة صفر يستمر في هذه الحلقة JP Z,HERE1;
 عند اختلاف الإشارة عن الصفر يزيد B بواحد INC B;
 HERE2: IN A,00 ; حلقة انتظار إلى أن ترجع الإشارة للصفر
 CP 01
 JP Z,HERE2
 يخرج محتويات المسجل B على بوابة الإخراج 00 ;
 يذهب للبداية ليقرأ نبضة جديدة JP HERE1;

شكل (5-10) برنامج المثال 5-6

جميع العمليات المنطقية تؤثر على الأعلام ما عدا علمي الحمل والحمل النصفي حيث يكونان دائما صفراء بعد أي عملية منطقية لأن الحمل والحمل النصفي غير معرف مع العمليات المنطقية .

الصورة العامة لأوامر العملية AND هي :

AND reg
 A ← المسجل A AND reg
 AND (HL)
 A ← المسجل A AND (HL)
 AND data8
 A ← المسجل A AND data8
 AND (IX+d)
 A ← المسجل A AND (IX+d)
 AND (IY+d)
 A ← المسجل A AND (IY+d)

بنفس الطريقة يمكن كتابة الصورة العامة لأوامر OR و XOR كما يلى :

OR reg
 OR (HL)
 OR data8
 OR (IX+d)

OR (IY+d)

XOR reg

XOR (HL)

XOR data8

XOR (IX+d)

XOR (IY+d)

هناك عملية NOT وهي لا تجري إلا على المسجل A حيث يقلب كل صفر إلى واحد وكل واحد إلى صفر ، والصورة العامة لهذا الأمر هي :

CPL

عكس المسجل \leftarrow المسجل A

هذه العملية تسمى عملية المتم الأحادي وهناك عملية المتم الثنائي المعرفة كالتالي :

المتم الثنائي = 1 + المتم الأحادي

وهناك أمر يقوم بعملية المتم الثنائي وصورته العامة هي :

NEG

المتم الثنائي للمسجل A \leftarrow المسجل

إن للمتم الثنائي أهمية خاصة في تحويل عمليات الطرح إلى جمع كما هو مشروح بالتفصيل في الملحق رقم 1 في نهاية الكتاب .

إلى هنا نكون قد انتهينا من العرض التفصيلي لمعظم أوامر الشريحة Z80 على أننا سنعرض هذه الأوامر أولاً في صورة مجموعات كما في الأشكال (5-11) ثم سنعرض الأوامر مرتبة ترتيباً أبجدياً كما في شكل (5-20).

LD	A,	B,	C,	D,	E,	H,	L,	(HL),	(IX+d),	(IY+d),
A	7F	47	4F	57	5F	67	6F	77	DD77dd	FD77dd
B	78	40	48	50	58	60	68	70	DD70dd	FD70dd
C	79	41	49	51	59	61	69	71	DD71dd	FD71dd
D	7A	42	4A	52	5A	62	6A	72	DD72dd	FD72dd
E	7B	43	4B	53	5B	63	6B	73	DD73dd	FD73dd
H	7C	44	4C	54	5C	64	6C	74	DD74dd	FD74dd
L	7D	45	4D	55	5D	65	6D	75	DD75dd	FD75dd
data8	3E	06	0E	16	1E	26	2E	36xx	DD36dd	FD36ddxx
	xx		xx							
(HL)	7E	46	4E	56	5E	66	6E			
(IX+d)	7E	46	4E	56	5E	66	6E			*
(IY+d)	7E	46	4E	56	5E	66	6E			**

* جميع شفرات أوامر هذا الصنف تسبقها DD ويعقبها dd مثلها مثل أوامر العمود.

** جميع شفرات أوامر هذا الصنف تسبقها FD ويعقبها dd مثلها مثل أوامر العمود.

XX يقصد بها البایت الثانية من الامر وهي data8

LD	BC,	DE,	HL,	SP,	IX,	IY,
(addr)	ED4B adr	ED5B adr	2A adr	ED7B adr	DD2A adr	FD2A adr
data16	01 dat16	11 dat16	21 dat16	31 dat16	DD21 dat16	FD21 dat16

LD (addr),	BC	DE	HL	SP	IX	IY
	ED43 adr	ED53 adr	22 adr	ED73 adr	DD22 adr	FD22 adr

	SR	BC	DE	HL	IX	IY
PUSH	F5	C5	D5	E5	DDE5	FDE5
POP	F1	C1	D1	E1	DDE1	FDE1

LD SP,	HL	IX	IY
	F9	DDF9	FDF9

شكل (11-5) مجموعة أوامر الانتقال للبروسيسور Z80

EX DE,HL	EB
EX (sp),HL	E3
EX (SP),IX	DDE3
EX (SP),IY	FDE3
EX SR,SR1	08
EXX	D9

شكل (12-5) مجموعة أوامر الاستبدال

	ADD A,	ADC A,	SUB	SBC A,	INC	DEC	CP
A	87	8F	97	9F	3C	3D	BF
B	80	88	90	98	04	05	B8
C	81	89	91	99	0C	0D	B9
D	82	8A	92	9A	14	15	BA
E	83	8B	93	9B	1C	1D	BB
H	84	8C	94	9C	24	25	BC
L	85	8D	95	9D	2C	2D	BD
(HL)	86	8E	96	9E	34	35	BE
data8	C6	CE	D6	DE			FE
	xx	xx	xx	xx			xx
(IX+d)	DD	DD	DD	DD	DD	DD	DD
	86	8E	96	9E	34	35	BE
	dd	dd	dd	dd	dd	dd	dd
(IY+d)	FD	FD	FD	FD	FD	FD	FD
	86	8E	96	9E	34	35	BE
	dd	dd	dd	dd	dd	dd	dd

شكل (13-5) مجموعة أوامر الحساب

	AND	OR	XOR
A	A7	B7	AF
B	A0	B0	A8
C	A1	B1	A9
D	A2	B2	AA
E	A3	B3	AB
H	A4	B4	AC
L	A5	B5	AD
(HL)	A6	B6	AE
data8	E6xx	F6xx	EExx
(IX+d)	DDA6dd	DDB6dd	DDAEdd
(IY+d)	FDA6dd	FDB6dd	FDAEdd

	CPL	NEG	CCF	SCF
A	2F	ED44	3F	37

شكل (14-5) مجموعة أوامر المنطق

	ADD HL,	ADC HL,	SBC HL,	ADD IX,	ADD IY,	INC	DEC
BC	09	ED 4A	ED 42	DD 09	FD 09	03	0B
DE	19	ED 5A	ED 52	DD 19	FD 19	13	1B
HL	29	ED 6A	ED 62			23	2B
SP	39	ED 7A	ED 72	DD39	FD 39	33	3B
IX				DD 29		DD 23	DD 2B
IY					FD 29	FD 23	FD 2B

شكل (15-5) أوامر حسابية على أزواج مسجلات

JP	JP Z	JP NZ	JP C	JP NC	JP M	JP P	JP PE	JP PO
C3	CA	C2	DA	D2	FA	F2	EA	E2
xx	xx	xx	xx	xx	xx	xx	xx	xx
xx	xx	xx	xx	xx	xx	xx	xx	xx

JR	JR Z	JR NZ	JR C	JR NC
18xx	28xx	20xx	38xx	30xx

CALL addr	CDxxxx
CALL Z,addr	CCxxxx
CALL NZ,addr	C4xxxx
CALL C,addr	DCxxxx
CALL NC,addr	D4xxxx
CALL M,addr	FCxxxx
CALL P,addr	F4xxxx
CALL PE,addr	ECxxxx
CALL PO,addr	E4xxxx

RET	C9
RET Z	C8
RET NZ	C0
RET C	D8
RET NC	D0
RET M	F8
RET P	F0
RET PE	E8
RET PO	E0

xxxx تمثل اثنين بايت للعنوان الذى سيتم القفز إليه
xx تمثل بايت واحدة للعنوان الذى سيتم القفز إليه فى حالة القفز النسبي

شكل (5-16) مجموعة أوامر القفز

الأمر BIT b,sss يجعل علم الصفر يساوى عكس البت رقم b فى المسجل أو
الذاكرة sss

BIT	0,	1,	2,	3,	4,	5,	6,	7,
A	CB 47	CB 4F	CB 57	CB 5F	CB 67	CB 6F	CB 77	CB 7F
B	CB 40	CB 48	CB 50	CB 58	CB 60	CB 68	CB 70	CB 78
C	CB 41	CB 49	CB 51	CB 59	CB 61	CB 69	CB 71	CB 79
D	CB 42	CB 4A	CB 52	CB 5A	CB 62	CB 6A	CB 72	CB 7A
E	CB 43	CB 4B	CB 53	CB 5B	CB 63	CB 6B	CB 73	CB 7B
H	CB 44	CB 4C	CB 54	CB 5C	CB 64	CB 6C	CB 74	CB 7C
L	CB 45	CB 4D	CB 55	CB 5D	CB 65	CB 6D	CB 75	CB 7D
(HL)	CB 46	CB 4E	CB 56	CB 5E	CB 66	CB 6E	CB 76	CB 7E
(IX+d)	DD CB d46	DD CB d4E	DD CB d56	DD CB d5E	DD CB d66	DD CB d6E	DD CB d76	DD CB d7E
(IY+d)	FD CB d46	FD CB d4E	FD CB d56	FD CB d5E	FD CB d66	FD CB d6E	FD CB d76	FD CB d7E

شكل (5-17) مجموعة اختبار و SET و RESET بت من برات مسجل أو مكان
فى الذاكرة

الأمر SET b,sss يجعل البت رقم b في المسجل أو الذاكرة sss تساوى واحد

SET	0,	1,	2,	3,	4,	5,	6,	7,
A	CB							
	C7	CF	D7	DF	E7	EF	F7	FF
B	CB							
	C0	C8	D0	D8	E0	E8	F0	F8
C	CB							
	C1	C9	D1	D9	E1	E9	F1	F9
D	CB							
	C2	CA	D2	DA	E2	EA	F2	FA
E	CB							
	C3	CB	D3	DB	E3	EB	F3	FB
H	CB							
	C4	CC	D4	DC	E4	EC	F4	FC
L	CB							
	C5	CD	D5	DD	E5	ED	F5	FD
(HL)	CB							
	C6	CE	D6	DE	E6	EE	F6	FE
(IX+d)	DD							
	CB							
	dC6	dCE	dD6	dDE	dE6	dEE	dF6	dFE
(IY+d)	FD							
	CB							
	dC6	dCE	dD6	dDE	dE6	dEE	dF6	dFE

تابع شكل (17-5) مجموعة اختبار و SET و RESET بت من برات مسجل أو مكان في الذاكرة

الأمر RES b,sss يجعل البت رقم b في المسجل أو الذاكرة sss تساوى صفر

RES	0,	1,	2,	3,	4,	5,	6,	7,
A	CB 87	CB 8F	CB 97	CB 9F	CB A7	CB AF	CB B7	CB BF
B	CB 80	CB 88	CB 90	CB 98	CB A0	CB A8	CB B0	CB B8
C	CB 81	CB 89	CB 91	CB 99	CB A1	CB A9	CB B1	CB B9
D	CB 82	CB 8A	CB 92	CB 9A	CB A2	CB AA	CB B2	CB BA
E	CB 83	CB 8B	CB 93	CB 9B	CB A3	CB AB	CB B3	CB BB
H	CB 84	CB 8C	CB 94	CB 9C	CB A4	CB AC	CB B4	CB BC
L	CB 85	CB 8D	CB 95	CB 9D	CB A5	CB AD	CB B5	CB BD
(HL)	CB 86	CB 8E	CB 96	CB 9E	CB A6	CB AE	CB B6	CB BE
(IX+d)	DD CB d86	DD CB d8E	DD CB d96	DD CB d9E	DD CB dA6	DD CB dAE	DD CB dB6	DD CB dB6
(IY+d)	FD CB d86	FD CB d8E	FD CB d96	FD CB d9E	FD CB dA6	FD CB dAE	FD CB dB6	FD CB dB6

تابع شكل (17-5) مجموعة اختبار و SET و RESET بت من بثات مسجل أو مكان في الذاكرة

	RLC	RRC	RL	RR	SLA	SRA	SRL
A	CB 07	CB 0F	CB 17	CB 1F	CB 27	CB 2F	CB 3F
B	CB 00	CB 08	CB 10	CB 18	CB 20	CB 28	CB 38
C	CB 01	CB 09	CB 11	CB 19	CB 21	CB 29	CB 39
D	CB 02	CB 0A	CB 12	CB 1A	CB 22	CB 2A	CB 3A
E	CB 03	CB 0B	CB 13	CB 1B	CB 23	CB 2B	CB 3B
H	CB 04	CB 0C	CB 14	CB 1C	CB 24	CB 2C	CB 3C
L	CB 05	CB 0D	CB 15	CB 1D	CB 25	CB 2D	CB 3D
(HL)	CB 06	CB 0E	CB 16	CB 1E	CB 26	CB 2E	CB 3E
(IX+d)	DD CB d06	DD CB d0E	DD CB d16	DD CB d1E	DD CB d26	DD CB d2E	DD CB d3E
(IY+d)	FD CB d06	FD CB d0E	FD CB d16	FD CB d1E	FD CB d26	FD CB d2E	FD CB d3E

أوامر خاصة بإزاحة أو دوران المسجل A فقط

RRA	RLA	RRCA	RLCA
1F	17	0F	07

شكل (18-5) مجموعة أوامر الإزاحة والدوران

IN	A,	B,	C,	D,	E,	H,	L,
Port No.	D8xx	-----	-----	-----	-----	-----	-----
(C)	ED78	ED40	ED48	ED50	ED58	ED60	ED68

OUT	Port No.,	(C),
A	D3xx	ED79
B	-----	ED41
C	-----	ED49
D	-----	ED51
E	-----	ED59
H	-----	ED61
L	-----	ED69

شكل (5-19) أوامر الإدخال والإخراج

لا تعمل شيء, NOP	00
توقف HALT	76
إخماد المقاطعة Disable Interrupt, DI	F3
تنشيط المقاطعة Enable Interrupt, EI	FB
تنشيط حالة المقاطعة رقم صفر IM0	ED46
تنشيط حالة المقاطعة رقم صفر IM1	ED56
تنشيط حالة المقاطعة رقم صفر IM2	ED5E

شكل (5-20) مجموعة أوامر متفرقة

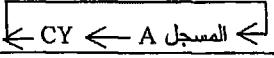
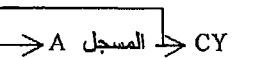
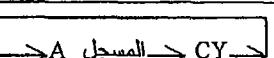
كانت هذه بعض أهم مجموعات الأوامر للمعالج Z80 والشائعة الاستخدام . شكل (5-21) يحتوى جميع أوامر الشريحة مرتبة ترتيباً أبجدياً مع نبذة عن ما يعملاه كل أمر وعدد نبضات الترددان الذى يأخذها (ن #) لكي يتم إحضاره من الذاكرة وتنفيذه والأعلام التى تتأثر بكل أمر وكذلك شفرة كل أمر حيث من هذه الشفرة يمكن استنتاج عدد بآيات الأمر . انظر الملاحظات الخاصة بشفرة الأوامر فى نهاية هذا الشكل .

شفرة الأسmbلي	الأعلام المتأثرة	# ن	شفرة الأمر	وظيفة الأمر
ADC A,reg	ZSP CY HC	4	10001sss	A ← A+CY+ reg
ADC A,(HL)	ZSP CY HC	7	8E	A ← A+CY+(HL)
ADC A,data8	ZSP CY HC	7	CE data8	A ← A+CY+data8
ADC A,(IY+d)	ZSP CY HC	19	FD 8E dd	A ← A+CY+(IY+d)
ADC A,(IX+d)	ZSP CY HC	19	DD 8E dd	A ← A+CY+(IX+d)
ADC HL, rp	-- CY -	15	ED 01rp1010	HL ← HL+CY+rp
ADD A,reg	ZSP CY HC	4	10000sss	A ← A + Reg
ADD A,(HL)	ZSP CY HC	7	86	A ← A + (HL)
ADD A,data8	ZSP CY HC	7	C6 data8	A ← A+data8
ADD A,(IY+d)	ZSP CY HC	19	FD 88 dd	A ← A+ (IY+d)
ADD A,(IX+d)	ZSP CY HC	19	DD 86 dd	A ← A+ (IX+d)
ADD HL, rp	-- CY -	11	00rp1001	HL ← HL+rp
ADD IY, rp	-- CY -	15	FD 00rp1001	IX ← IY + rp
ADD IX, rp	-- CY -	15	DD 00rp1001	IX ← IX + rp
AND reg	ZS P 0 0	4	10100xxx	A ← A AND reg
AND (HL)	ZS P 0 0	7	A6	A ← A AND (HL)
AND data8	ZS P 0 0	7	E6 data8	A ← A AND data8
AND (IY+d)	ZS P 0 0	19	FD A6 dd	A ← A AND (IY+d)
AND (IX+d)	ZS P 0 0	19	DD A6 dd	A ← A AND (IX+d)
BIT b,reg	Z - - - -	9	CB 01bbbsss	ZF ← reg عكس البت b في
BIT b,(HL)	Z - - - -	12	CB 01bbb110	ZF ← (HL) عكس البت b في (HL)
BIT b,(IY+d)	Z - - - -	20	FD CB dd 01bbb110	ZF ← (IY+D) عكس البت b في (IY+D)
BIT b,(IX+d)	Z - - - -	20	DD CB dd 01bbb110	ZF ← (IX+D) عكس البت b في (IX+D)
CALL addr	-- - - - -	17	CD addr	نداء غير مشروط لبرنامج فرعى
CALL C,addr	-- - - - -	10/17	DC addr	نداء مشروط بعلم الحمل =1
CALL M,addr	-- - - - -	10/17	FC addr	نداء مشروط بعلم إشارة =1
CALL NC,addr	-- - - - -	10/17	D4 addr	نداء مشروط بعلم الحمل =0
CALL NZ,addr	-- - - - -	10/17	C4 addr	نداء مشروط بعلم الصفر =0
CALL P,addr	-- - - - -	10/17	F4 addr	نداء مشروط بعلم إشارة =0
CALL PE,addr	-- - - - -	10/17	EC addr	نداء مشروط بعلم باريتي =1
CALL PO,addr	-- - - - -	10/17	E4 addr	نداء مشروط بعلم باريتي =0
CALL Z,addr	-- - - - -	10/17	CC addr	نداء مشروط بعلم الصفر =1
CCF	-- CY HC	4	3F	اعكس علم الحمل
CP reg	ZSP CY HC	4	10111sss	A - reg مقارنة
CP (HL)	ZSP CY HC	7	BE	A - (HL) مقارنة (HL)
CP const	ZSP CY HC	7	FE data8	A - const مقارنة const
CP (IY+d)	ZSP CY HC	19	FD BE data8	A - (IY+d) مقارنة (IY+d)
CP (IX+d)	ZSP CY HC	19	DD BE data8	A - (IX+d) مقارنة (IX+d)
CPL	- - - - -	4	2F	اعكس المسجل A

CPI	ZSP CY HC	16	ED A1	$\Delta-(HL)$ BC \leftarrow BC-1 , HL \leftarrow HL+
CPJR	ZSP CY HC	21/16	ED B1	BC=0 إلى CPI
CPD	ZSP CY HC	16	ED A9	مقارنة (HL) BC \leftarrow BC-1 , HL \leftarrow HL-
CPDR	ZSP CY HC	21/16	ED B9	BC=0 إلى CPT
DAA	ZSP CY HC	4	27	حول المركم للنظام العشري
DEC reg	ZSP - HC	4	00ddd101	reg \leftarrow reg -
DEC (HL.)	ZSP - HC	7	35	(HL.) \leftarrow (HL.) - 1
DEC (IY+d)	ZSP - HC	19	FD 35 data8	(IY+d) \leftarrow (IY+d) -
DEC (IX+d)	ZSP - HC	19	DD 35 data8	(IX+d) \leftarrow (IX+d) -
DI	-----	4	F3	أحمل المقطعة IF \leftarrow 0
DEC rp	-----	6	00rp1011	rp \leftarrow rp - 1
DEC IY	-----	10	FD 2B	IY \leftarrow IY - 1
DEC IX	-----	10	DD 2B	IX \leftarrow IX - 1
DJNZ ddd	-----	8/13	10 ddd	قف بمقدار ddd و إقصاص B بمقدار 1 إلى أن يصبح B=0
EI	-----	4	F3	اسمح بالمقاطعة IF \leftarrow
EX DE,HL	-----	4	E3	IIL \leftarrow DE
EX AF,AF1	-----	4	08	PSW1 \leftarrow PSW
EXX	-----	4	D9	BCDEIIL1 \leftrightarrow BCDEIIL
EX (SP),HL	-----	19	E3	(SP+1) \rightarrow H (SP) \rightarrow L
EX (SP),IX	-----	23	FD E3	(SP+1) \rightarrow IY/H (SP) \rightarrow IY/L
EX (SP),IX	-----	23	DD E3	(SP+1) \rightarrow IX/H (SP) \rightarrow IX/L
HALT	-----	4	76	أوقف تنفيذ البرنامج
IM 0	-----	8	ED 46	حالة المقطعة صفر
IM 1	-----	8	ED 56	حالة المقطعة واحد
IM 2	-----	8	ED 5E	حالة المقطعة اثنين
IN A,(no.)	-----	11	DB no8	البوابة رقم A \leftarrow no.
IN reg,(C)	-----	12	ED 01ddd000	البوابة reg \leftarrow (C)
INI	-----	16	ED A2	port (C) \rightarrow (HL) B-1 \rightarrow B HL \leftarrow IIL+1
INJR	-----	16/21	ED B2	كرر INI إلى أن B=0
IND	-----	16	ED AA	port(C) \rightarrow (HL) B-1 \rightarrow B HL \leftarrow HL-1
INDR	-----	16/21	ED BA	كرر IND إلى أن B=0
INC reg	Z SP - HC	4	00ddd100	reg \leftarrow reg + 1
INC (HL)	Z SP - HC	7	34	(HL) + 1 \rightarrow (HL)
INC (IY+d)	Z SP - HC	19	FD 34 dd	(IY+d) \leftarrow (IY+d) + 1
INC (IX+d)	Z SP - HC	19	DD 34 dd	(IX+d) \leftarrow (IX+d) + 1

INC rp	- - - - -	6	00xx0011	rp \leftarrow rp + 1
INC IY	- - - - -	10	FD 23	IY \leftarrow IY + 1
INC IX	- - - - -	10	DD 23	IX \leftarrow IX + 1
JP addr	- - - - -	10	C3 addr	قفز غير مشروط
JP (HL)	- - - - -	4	E9	قفز إلى عنوان في HL
JP (IX)	- - - - -	8	DD E9	قفز إلى عنوان في IX
JP (IY)	- - - - -	8	FD E9	قفز إلى عنوان في IY
JP Z,addr	- - - - -	10	CA addr	قفز مشروط بعلم الصفر = 1
JP NZ,addr	- - - - -	10	C2 addr	قفز مشروط بعلم الصفر = 0
JP C,addr	- - - - -	10	DA addr	قفز مشروط بعلم الحمل = 1
JP NC,addr	- - - - -	10	D2 addr	قفز مشروط بعلم الحمل = 0
JP PO,addr	- - - - -	10	E2 addr	قفز مشروط بعلم باريتي = 0
JP PE,addr	- - - - -	10	EA addr	قفز مشروط بعلم باريتي = 1
JP P,addr	- - - - -	10	F2 addr	قفز مشروط بعلم اشارة = 0
JP M,addr	- - - - -	10	FA addr	قفز مشروط بعلم اشارة = 1
JR dd	- - - - -	10	18 dd	قفز نسبي غير مشروط
JR Z,dd	- - - - -	7/12	28 dd	Z=1
JR NZ,dd	- - - - -	7/12	20 dd	Z=0
JR C,dd	- - - - -	7/12	38 dd	CY=1
JR NC,dd	- - - - -	7/12	30 dd	CY=0
LD reg1,reg2	- - - - -	4	01dddsss	reg2 \rightarrow reg1
LD reg,(HL)	- - - - -	7	01ddd110	(HL) \rightarrow reg
LD (HL),reg	- - - - -	7	01110sss	(HL) \leftarrow reg
LD reg,data8	- - - - -	7	11ddd110 data8	reg \leftarrow data8
LD reg,(IY+d)	- - - - -	19	FD 01ddd110 dd	reg \leftarrow (IY+d)
LD reg,(IX+d)	- - - - -	19	DD 01ddd110 dd	reg \leftarrow (IX+d)
LD (IY+d),reg	- - - - -	19	FD 01110sss dd	(IY+d) \leftarrow reg
LD (IX+d),reg	- - - - -	19	DD 01110sss dd	(IX+d) \leftarrow reg
LD (HL),data8	- - - - -	10	36 data8	(HL) \leftarrow data8
LD (IY+d),data8	- - - - -	19	FD 36 dd data8	(IY+d) \leftarrow data8
LD (IX+d),data8	- - - - -	19	DD 36 dd data8	(IX+d) \leftarrow data8
LD A,(addr)	- - - - -	13	3A addr	A \leftarrow (addr)
LD (addr),A	- - - - -	13	32 addr	(addr) \leftarrow A
LD (addr),BC	- - - - -	20	ED 43 addr	addr \leftarrow C addr+1 \leftarrow B
LD (addr),DE	- - - - -	20	ED 53 addr	addr \leftarrow E addr+1 \leftarrow D
LD (addr),HL	- - - - -	20	22 addr	addr \leftarrow L addr+1 \leftarrow H
LDD (addr),IX	- - - - -	20	DD 22 addr	addr \leftarrow IX/L addr+1 \leftarrow IX/H
LDD (addr),IY	- - - - -	20	FD 22 addr	addr \leftarrow IY/L addr+1 \leftarrow IY/H
LDD (addr),SP	- - - - -	20	ED 73 addr	addr \leftarrow SP/L addr+1 \leftarrow SP/H
LD A,(BC)	- - - - -	7	0A	A \leftarrow (BC)
LD A,(DE)	- - - - -	7	1A	A \leftarrow (DE)
LD (BC),A	- - - - -	7	02	A \rightarrow (BC)
LD (DE),A	- - - - -	7	12	A \rightarrow (DE)
LD A,I	- - - - -	9	ED 57	A \leftarrow I
LD I,A	- - - - -	9	ED 47	I \leftarrow A

LD A,R	- - -	9	ED 5F	A ← R
LD R,A	- - -	9	ED 4F	R ← A
LD rp,data16	- - -	10	00rp0001 data16	rp ← data16
LD IX,data16	- - -	14	DD 21 data16	IX ← data16
LD rp,(addr)	- - -	20	ED 01rp1011 addr	(addr+1) → B (addr) → C
LD HL,(addr)	- - -	16	2A addr	(addr+1) → H (addr) → L
LD IX,(addr)	- - -	20	DD 2A addr	(addr) → IX/L IX/H ← (addr+1),
LD IY,(addr)	- - -	20	FD 2A addr	(addr) → IY/L IY/H ← (addr+1),
LD SP,HL	- - -	16	F9	HL → SP
LD SP,IX	- - -	10	DD F9	SP ← IX
LD SP,IY	- - -	10	FD F9	SP ← IY
LDI	Z S P - -	16	ED A0	DE ← DE+1 (HL) → (DE) HL ← HL+1 BC ← BC-1
LDIR	Z S P - -	21/16	ED B0	BC=0 نفس الأمر السابق إلى
LDD	Z S P - -	16	ED AB	DE ← DE-1 (HL) → (DE) BC ← BC-1 HL ← HL-1
LDDR	Z S P - -	21/16	ED BB	BC=0 نفس الأمر السابق إلى
NIEG	Z S P - -	8	ED 44	A ← ~
NOP	- - -	4	00	No operation لا تعمل شيئاً
OR reg	Z S P 0 0	4	I0110sss	A ← A OR reg
OR (HL)	Z S P 0 0	7	B5	A ← A OR (HL)
OR data8	Z S P 0 0	7	F6 data8	A ← A OR data8
OR (IY+d)	Z S P 0 0	19	FD B6 dd	A ← A OR (IY+d)
OR (IX+d)	Z S P 0 0	19	DD B6 dd	A ← A OR (IX+d)
OUT (no.),A	- - -	11	D3 no.8	port (no.) ← A
OUT (C),reg	- - -	12	ED 01sss001	port (C) ← reg
OUTI	- - -	16	ED A3	(HL) → port(C) B-1 → B HL → HL+1
OTIR	- - -	16/21	ED B3	B=0 كرر OUTI إلى أن
OUTD	- - -	16	ED A8	B ← B-1, port(C) ← (HL) HL ← HL-1
OTDR	- - -	16/21	ED B8	B=0 كرر OUTD إلى أن
PUSH rp	- - -	11	11rp0101	المسجلان rp ← قمة المكذبة
PUSH IY	- - -	15	FD E5	المسجل IY ← قمة المكذبة
PUSH IX	- - -	15	DD E5	المسجل IX ← قمة المكذبة
POP rp	- - -	11	11rp0001	rp ← المسجلين قمة المكذبة
POP IY	- - -	15	FD E1	IY ← المسجل قمة المكذبة
POP IX	- - -	15	DD E1	IX ← المسجل قمة المكذبة
RLCA	- - CY -	4	07	CY ← A المسجل ←

RLA	- - CY -	4	17	
RRCA	- - CY -	4	0F	
RRA	- - CY -	4	1F	
RLC reg	- - CY -	8	CB 0000sss	دوران مسجل ، مثل RLCA
RLC (HL)	- - CY -	15	CB 06	دوران عنوان في HL مثل RLCA
RLC (IY+d)	- - CY -	23	FD CB dd 06	دوران عنوان(IY+d) مثل RLCA
RLC (IX+d)	- - CY -	23	DD CB dd 06	دوران عنوان(IX+d) مثل RLCA
RL (HL)	- - CY -	15	CB 16	دوران (HL) لليسار من خلال علم الحمل
RL (IX+d)	- - CY -	23	DD CB dd 16	دوران (IX+d) لليسار من خلال علم الحمل
RL (IY+d)	- - CY -	23	FD CB dd 16	دوران (IY+d) لليسار من خلال علم الحمل
RL reg	- - CY -	8	CB 00010sss	دوران reg لليسار من خلال علم الحمل
RLD	- - - -	18	ED 6F	دوران بيت الأولى من A للشمال مع العنوان الموجود في HL
RRD	- - - -	18	ED 67	دوران بيت الأولى من A لليمين مع العنوان الموجود في HL
RR (HL)	- - CY -	15	CB 1E	دوران (HL) لليمين من خلال علم الحمل
RR (IX+d)	- - CY -	23	DD CB dd 1E	دوران (IX+d) لليمين من خلال علم الحمل
RR (IY+d)	- - CY -	23	FD CB dd 1E	دوران (IY+d) لليمين من خلال علم الحمل
RR reg	- - CY -	8	CB 00011sss	دوران reg لليمين من خلال علم الحمل
RRC (HL)	- - CY -	15	CB 0E	دوران (HL) لليمين مثل RRCA
RRC (IX+d)	- - CY -	23	DD CB dd 0E	دوران (IX+d) لليمين مثل RRCA
RRC (IY+d)	- - CY -	23	FD CB dd 0E	دوران (IY+d) لليمين مثل RRCA
RRC reg	- - CY -	8	CB 00001sss	دوران reg لليمين مثل RRCA
RET	- - - -	10	C9	عودة من برنامج فرعي
RET Z	- - - -	5/11	C8	عودة مشروطة بعلم الصفر = 1
RET NZ	- - - -	5/11	C0	عودة مشروطة بعلم الصفر = 0
RET C	- - - -	5/11	D8	عودة مشروطة بعلم الحمل = 1
RET NC	- - - -	5/11	D0	عودة مشروطة بعلم الحمل = 0
RET PO	- - - -	5/11	E0	عودة مشروطة بعلم باريتي = 0
RET PE	- - - -	5/11	E8	عودة مشروطة بعلم باريتي = 1
RET M	- - - -	5/11	F8	عودة مشروطة بعلام إشاره = 1
RET P	- - - -	5/11	F0	عودة مشروطة بعلم إشاره = 0
RETI	- - - -	14	ED 4D	عودة من مقاطعة
RETN	- - - -	14	ED 45	عودة من مقاطعة ذات قناع

RES b,reg	-----	8	CB 10bbbbss	صفر في البت b في reg
RFS b,(HL)	-----	15	CB 10bbb110	صفر في البت b في (HL)
RES b,(IX+d)	-----	23	DD CB dd 10bbb110	صفر في البت b في (IX+d)
RES b,(IY+d)	-----	23	FD CB dd 10bbb110	صفر في البت b في (IY+d)
RST n	-----	11	11nnnn111	إعادة تشغيل
SUB reg	Z S P CY HC	4	10010sss	A ← A - reg
SUB (HL)	Z S P CY HC	7	96	A ← A - (HL)
SUB data8	Z S P CY HC	7	D6 data8	A ← A - data8
SUB (IX+d)	Z S P CY HC	19	DD 96 dd	A ← A - (IX+d)
SUB (IY+d)	Z S P CY HC	19	FD 96 dd	A ← A - (IY+d)
SBC reg	Z S P CY HC	4	10011sss	A ← A - CY - reg
SBC (HL)	Z S P CY HC	7	9E	A ← A - CY - (HL)
SBC data8	Z S P CY HC	7	DE data8	A ← A - CY - data8
SBC (IX+d)	Z S P CY HC	19	DD 9E dd	A ← A - CY - (IX+d)
SBC (IY+d)	Z S P CY HC	19	FD 9E dd	A ← A - CY - (IY+d)
SBC HL,rp	Z S P CY HC	15	ED 01rp0010	HL ← HL - CY - rp
SLA (HL)	Z S P CY HC	2	CB 26	إزاحة (HL) لليسار ، حمل 0 في بت 0
SLA (IX+d)	Z S P CY HC	4	DD CB dd 26	إزاحة (IX+d) لليسار ، حمل 0 في بت 0
SLA (IY+d)	Z S P CY HC	4	FD CB dd 26	إزاحة (IY+d) لليسار ، حمل 0 في بت 0
SLA reg	Z S P CY HC	2	CB 00100sss	إزاحة reg لليسار حمل 0 في بت 0
SRL (HL)	Z S P CY HC	2	CB 3E	إزاحة (HL) لليمين ، حمل 0 في بت 7
SRL (IX+d)	Z S P CY HC	4	DD CB dd 3E	إزاحة (IX+d) لليمين ، حمل 0 في بت 7
SRL (IY+d)	Z S P CY HC	4	FD CB dd 3E	إزاحة (IY+d) لليمين ، حمل 0 في بت 7
SRL reg	Z S P CY HC	2	CB 00111sss	إزاحة reg لليمين ، حمل 0 في بت 7
SRA (HL)	Z S P CY HC	2	CB 2E	إزاحة (HL) لليمين ، بت 7 تظل كما هي
SRA (IX+d)	Z S P CY HC	4	DD CB dd 2E	إزاحة (IX+d) لليمين ، بت 7 تظل كما هي
SRA (IY+d)	Z S P CY HC	4	FD CB dd 2E	إزاحة (IY+d) لليمين ، بت 7 تظل كما هي
SRA reg	Z S P CY HC	2	CB 00101sss	إزاحة reg لليمين بت 7 تظل كما هي
SET b,reg	Z S P CY HC	8	CB 11bbbbss	واحد في البت b في reg
SET b,(IY+d)	Z S P CY HC	23	FD CB dd 11bbb110	واحد في البت b في (IY+d)
SET b,(HL)	Z S P CY HC	15	DD CB dd 11bbb110	واحد في البت b في (HL)

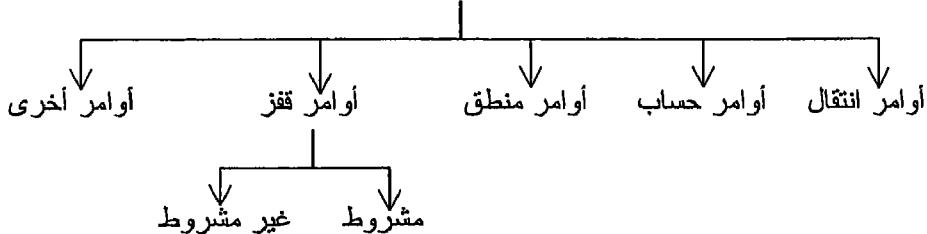
SET b,(IX+d)	Z S P CY HC	23	DD CB dd 11bbb110	واحد في البت b في (IX+d)
XOR reg	Z SP 0 0	4	10101sss	A ← A XOR reg
XOR (HL)	Z SP 0 0	7	AE	A ← A XOR (HL)
XOR data8	Z SP 0 0	7	EE data8	A ← A XOR data8
XOR (IY+d)	Z SP 0 0	19	FD AE dd	A ← A XOR (IY+d)
XOR (IX+d)	Z SP 0 0	19	DD AE dd	A ← A XOR (IX+d)

شكل (21-5) أوامر الشريحة Z80 مرتبة أبجديا

- sss أو dd تستبدل بشفرة مسجل 8 بت كما في فصل 2 .
- rp تستبدل بشفرة زوج مسجلات كما في فصل 2 .
- data8 ثابت مكون من 8 بت (بايت) .
- إزاحة عنوانية مكونة من 8 بت (بايت) .
- dd رقم بت معينة في مسجل أو بايت ذاكرة .
- bbb رقم (عنوان) بوابة (إدخال أو إخراج) من 8 بت (بايت) .

10-5 تمارين

1. أكمل الجدول التالي الخاص بأوامر الشريحة Z80 :
أوامر الشريحة Z80



2. ما هي نتيجة تنفيذ البرنامج التالي :

E000 LD A,05
E002 LD B,A
E003 LD C,B
E004 LD D,C
E005 LD E,D
E006 LD L,E
E007 LD H,L
E008 LD (HL),L

3. اقرأ البرنامج السابق وأجب عما يلى :
• محتويات مكان الذاكرة = E000

.4

- محتويات مكان الذاكرة E001=.....
- محتويات مكان الذاكرة 0505=.....

E000 LD HL,E100
E003 LD (HL),3E
E005 INC HL
E006 LD (HL),05
E008 INC HL
E009 LD (HL),47
E00B INC HL
E00C LD (HL),48

ما هي نتيجة تنفيذ البرنامج السابق ؟

5. على ضوء نتيجة تنفيذ البرنامج السابق ما هي نتيجة تنفيذ الشفرات الموجودة في الأماكن E100 إلى E103 ؟

6. هل تتأثر الأعلام بأوامر الانتقال ؟

7. أذكر الأعلام التي تتأثر بكل عملية من العمليات الحسابية والمنطقية ؟

8. إذا كانت محتويات المسجل A=F3H و محتويات المسجل B=A4H فاكتب محتويات المسجل A بعد تنفيذ كل أمر من الأوامر التالية على نفس المحتويات السابقة ووضح أيضاً كيف ستتأثر الأعلام بكل أمر :

ADD A,B
SUB A,B
SUB A,A
INC A
AND B
OR B
XOR B

9. ارسم مخطط السيير للبرنامج التالي وما هي نتيجة تنفيذه :

E000 LD L,50H
E003 LD H,E1H
E005 LD (HL),A
E006 DEC L
E007 JPNZ E005

10. ماذا يحدث لو كتبنا البرنامج السابق عند E100 بدلاً من E000 ؟

11. أعد كتابة البرنامج السابق مستخدماً العلامات Labels ؟ وما هي مميزات البرنامج مكتوباً بهذه الصورة ؟

12. اكتب برنامجاً يحسب عدد الوحدات الموجودة في محتويات المسجل A ، مثلاً إذا كان A=11110101 فإن عدد الوحدات = 6 .

13. كم عدد بوابات الإدخال التي يستطيع البروسيسور Z80 التعامل معها؟
14. كم عدد بوابات الإخراج التي يستطيع البروسيسور Z80 التعامل معها؟
15. على ماذا يتوقف هذا العدد ؟
16. هل هناك ما يمنع أن تكون بوابتي إدخال وإخراج لهما نفس الرقم ، كمثال على ذلك IN 05 و OUT 05 ؟
17. OUT (C),reg هذا أحد أوامر الإخراج للبروسيسور Z80 والذى يعني أخراج محتويات المسجل reg على بوابة الإخراج التى رقمها فى المسجل C فهل البروسيسور 8085 لديه ما يكفىء هذا الأمر ؟
18. هل تتأثر الأعلام بأوامر الإدخال والإخراج ؟
19. أكتب برنامجا يقرأ محتويات البوابة 00 وإذا كانت هذه المحتويات زوجية يخزنها فى الذاكرة ابتداء من العنوان E100 وإذا كانت فردية يخرجها على البوابة 00 ؟
20. اذكر طرق العنونة memory addressings المستخدمة مع البروسيسور Z80 والأوامر المستخدمة مع كل طريقة ؟ ومتى يفضل استخدام كل طريقة ؟
21. اكتب برنامج يحسب أكبر قيمة عدديه فى بايت فى المدى العنوانى E200H إلى E250H .
22. اكتب برنامج يحسب عدد البيانات التى تحتوى أصفرا والتى تحتوى أرقاما موجبة والنى تحتوى أرقاما سالبة فى المدى العنوانى E100 إلى E150 .
23. اكتب برنامج يحسب عدد البيانات التى تحتوى بيانات فردية والتى تحتوى بيانات زوجية فى المدى العنوانى E100 إلى E150 .
24. المدى العنوانى E100 إلى E150 يحتوى بيانات لإشارة صوت ، احسب كم مرة عبرت إشارة الصوت الصفر .
25. اكتب برنامج يقرأ بوابة الإدخال رقم 00 ويختبر البت الرابعة فيها ، فإذا كانت هذه البت واحد يخرج هذه المحتويات على البوابة 00 ، وإذا كانت هذه البت صفر يخرج محتوياتها على البوابة 01 .
26. اكتب برنامج يقرأ بوابة الإدخال رقم 00 إلى مالانهاية ويختبر البيانات التى يقرأها ، فإن كانت فردية يخرجها على البوابة 00 ، وإن كانت زوجية يخرجها على البوابة 01 . احسب أكبر معدل لدخول البيانات لكي يعمل هذا النظام فى الزمن المباشر real time .

الفصل السادس

المعالج

من البداية ... حتى النهاية

*Microprocessor... from Start ...
to end*

6-1 مقدمة

سنقوم في هذا الفصل بعملية بناء تدريجية لمعالج افتراضي يقوم بعدد محدود من العمليات الحسابية والمنطقية وله عدد محدود من الأوامر كما أن له عدداً محدوداً جداً من الخطوط في مسارات البيانات والعناوين والتحكم ولذلك فإن هذا المعالج يستطيع التعامل مع كمية محدودة جداً من بيانات الذاكرة وستبدأ عملية البناء من أقل مستوى ممكن ثم سنرتقي بها خطوة بخطوة إلى أن نصل إلى معالج متكامل ولكن بالمواصفات التي ذكرناها سابقاً . من خلال عملية البناء سنتعرف على وحدة الحساب والمنطق وكيفية عملها . ولقد كان قصتنا من وضع هذا الفصل في هذا الترتيب أن يكون القارئ قد ألم بفكرة عامة عن تركيب المعالج من الفصول السابقة ثم يجيء هذا الفصل فيؤكّد هذه الفكرة ويمحضها ويضيف إليها التفاصيل الدقيقة التي قد تجيب على الكثير من الأسئلة التي تدور في خلد أي قارئ عن كيفية تنفيذ أي معالج لأى أمر وما الذي يتحكم في عدد أوامره وغير ذلك من الأسئلة المهمة .

إن وحدة الحساب والمنطق هي إحدى المكونات الرئيسية للمعالج و مهمتها الأساسية هي إجراء العمليات الحسابية والمنطقية الأساسية وستبدأ فيما يلى عملية بناء هذه الوحدة ولكن نصل إلى ذلك لابد أن نتعرّف أولاً على كيفية تنفيذ عمليات الجمع والطرح في النظام الثنائي .

6-2 الجمع الثنائي Binary addition

مثال 1-6

أوجد ناتج جمع الرقم $B=b3b2b1b0 =1011$ مع الرقم $A=a3a2a1a0 =1101$ مع الرسم إن عملية الجمع تتم كالتالي :

$$\begin{array}{r} & 1 & 1 & 1 & 1 & \text{الحمل} \\ A & 1 & 1 & 0 & 1 \\ B & 1 & 0 & 1 & 1 \\ \hline & 1 & 1 & 0 & 0 & 0 \end{array}$$

↑ حمل

S3 S2 S1 S0 النتيجة

كما نرى فإن عملية الجمع تتم على عدد من المراحل ، المرحلة الأولى هي جمع a_0 (البت الأولى في الرقم A) مع b_0 (البت الأولى من الرقم B) فينتج من ذلك ناتج $s_0=0$ وحمل $c_0=1$ إلى المرحلة التالية . في المرحلة الثانية يتم جمع البتات الآتية :

$$c_0 + b_1 + a_1$$

حيث c_0 هي الحمل من المرحلة السابقة كما ذكرنا . نتيجة جمع المرحلة الثانية ستكون $c_1=0$ والحمل منها سيكون $c_1=1$ إلى المرحلة التالية . في المرحلة الثالثة ستم عملية الجمع التالية :

$$c_1 + b_2 + a_2$$

وسينتج عنها $s_2=0$ و $c_2=1$ ثم في مرحلة الجمع الرابعة سيتم جمع البثات التالية:

$$c_2 + b_3 + a_3$$

وسينتج عنها $s_3=1$ و $c_3=1$ وبذلك تنتهي عملية الجمع ويتبقي حمل آخر للخانة الرابعة وهو $c_4=1$ الذي سنهمله الآن . من ذلك نرى أننا في المرحلة الأولى نجمع اثنين بت فقط $(a_0 + b_0)$ وأما في باقي المراحل فإننا نجمع ثلاثة بثات $(a_n + c_{n-1} + b_n)$ حيث a_n هي البت رقم n في العدد A و b_n هي البت رقم n أيضا في العدد B ، وأما c_{n-1} فهي الحمل الناتج من جمع المرحلة السابقة $(c_{n-2} + b_{n-1} + a_{n-1})$. الآن نريد تكوين دائرة منطقية تقوم بعملية جمع 2 بت وأخرى تقوم بعملية جمع 3 بثات ثم من الدائرتين تقوم بتكوين دائرة تجمع العددين A و B .

6-2-1 دائرة نصف المجمع Half adder circuit, HA

يقوم نصف المجمع Half Adder, HA بجمع اثنين بت a_0 و b_0 ويعطى في الخرج النتيجة s_0 وحمل c_0 ويبين شكل (6-1) جدول الحقيقة truth table لهذه الدائرة . من جدول الحقيقة نستطيع كتابة المعادلات المنطقية التالية لكل من خرجي دائرة نصف المجمع :

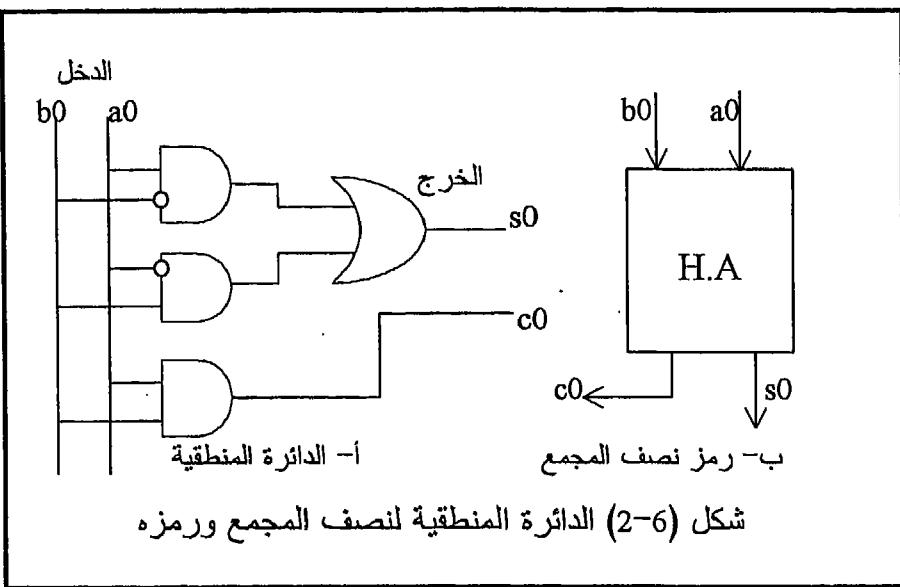
$$s_0 = a_0 \overline{b_0} + \overline{a_0} b_0 \quad 1-6$$

$$c_0 = a_0 b_0 \quad 2-6$$

الدخل		الخرج	
b_0	a_0	s_0	c_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

شكل (6-1) جدول الحقيقة لنصف المجمع

من المعادلتين 6-1 و 6-2 نستطيع رسم دائرة منطقية لنصف المجمع كما في الشكل (6-2أ) . انظر أيضا في نفس الشكل إلى الرمز الذي سنستخدمه لهذه الدائرة .



6-2-2 دائرة المجمع الكامل Full adder, FA

دائرة المجمع الكامل تكون قادرة على جمع ثلاثة بิตات (a_{n-1}, b_n, a_n) وينتج منها المجموع s_n والحمل للمرحلة القادمة c_n . جدول الحقيقة لهذه الدائرة موضح في شكل (3-6). من جدول الحقيقة نستطيع كتابة معادلات الخرج كما يلى :

$$S_n = a_n \bar{b}_n \bar{c}_{n-1} + \bar{a}_n b_n \bar{c}_{n-1} + \bar{a}_n \bar{b}_n c_{n-1} + a_n b_n c_{n-1} \quad 3-6$$

$$C_n = a_n b_n \bar{c}_{n-1} + a_n \bar{b}_n c_{n-1} + \bar{a}_n b_n c_{n-1} + a_n b_n c_{n-1} \quad 4-6$$

من المعادلتين 3-6 و 4-6 نستطيع استنتاج الدائرة المنطقية للمجمع الكامل كما في شكل (6-4). من ذلك نرى أنه لجمع أي رقمين A و B فإننا سنحتاج لنصف مجمع لجمع البت رقم 0 في كل من الرقمين ثم سنحتاج مجمعاً كاملاً لجمع كل بت في الرقم الأول مع ما يناظرها في الرقم الثاني مع الحمل الناتج من عملية الجمع السابقة. فمثلاً لو أن الرقمين A و B كل منهما يتكون من 4 بิตات فإننا سنحتاج إلى نصف مجمع وثلاثة مجتمعات كاملة لإتمام عملية جمع الرقمين ونفس الكلام يمكن تطبيقه على عملية جمع أي رقمين حيث كل منهما مكون من أي عدد من البتات. شكل (6-5) يبين الدائرة المستخدمة لجمع رقمين كل منهما مكون من أربعة بิตات كمثال على ذلك. وترى في نفس الشكل الرمز العام المستخدم للمجمع.

الدخل			الخرج	
cn-1	bn	an	sn	cn
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

شكل (6-3) جدول الحقيقة للمجمع الكامل

3-6 الطرح الثنائي Binary subtraction

لإجراء عمليات الطرح الثنائي فإنه عادة ما نلجأ إلى تحويل عملية الطرح إلى عملية جمع وبعد ذلك يمكن استخدام المجمع الذي سبق شرحه لتنفيذ عملية الطرح . لتحويل عملية الطرح إلى عملية جمع ننظر إلى المثال التالي :

مثال 2-6

اففترض أن لدينا الرقم $A = 1101$ فإن المعكوس أو المتمم الأحادي's complement لهذا الرقم هو 0010 وتم ذلك عن طريق قلب كل 1 إلى 0 وكل 0 إلى 1 في الرقم الأصلي . الآن ماذا يحدث لو جمعنا العدد الأصلي زائد متممه الأحادي زائد واحد كما يلى : ..

$$\begin{array}{r}
 A = 1101 \\
 \overline{A} = 0010 \\
 \hline
 1 +
 \end{array}$$

1 0000 ← الحمل

إن النتيجة كما رأينا ستكون دائماً صفرًا مع حمل واحد ، ولذلك فإنه بإهمال هذا الحمل يمكننا كتابة العلاقة التالية :

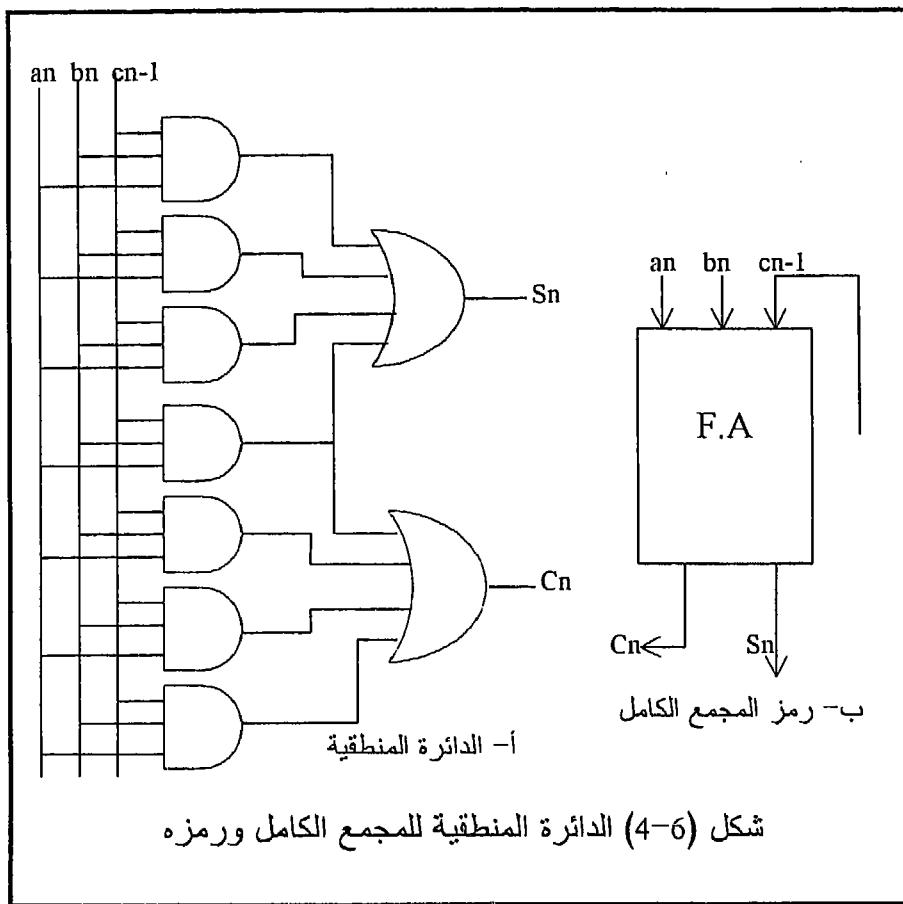
$$A + \overline{A} + 1 = 0$$

ومنها يمكن كتابة الرقم A على الصورة التالية :

$$-A = \overline{A} + 1$$

5-6

وعلى ذلك فإنه من المعادلة (5-6) يمكننا أن نرى أن أي عملية طرح يمكن تحويلها إلى عملية جمع عن طريق استبدال المطروح بمتممه الثنائي (المتكم الأحادي + 1) . كمثال على ذلك انظر إلى عمليات الطرح التالية وكيف حولناها إلى عمليات جمع :

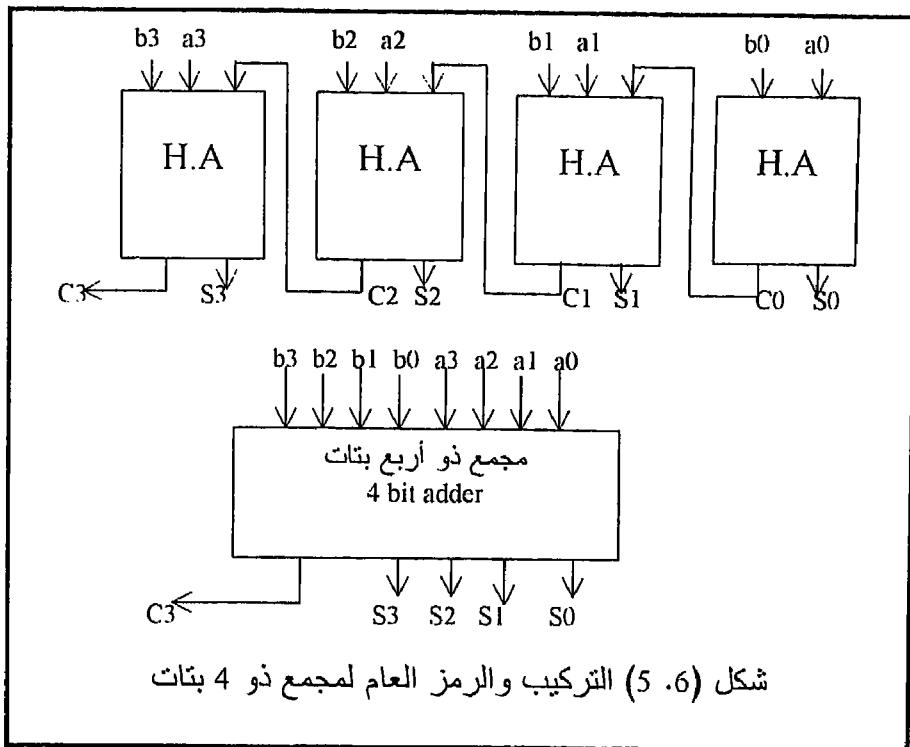


$$A - B = A + \overline{B} + 1 \quad 6-6$$

$$B - C = B + \overline{C} + 1 \quad 7-6$$

وبذلك نستطيع القول أنه يمكننا استخدام دائرة المجمع التي سبق شرحها في تنفيذ عمليات الطرح أيضاً بعد إجراء بعض التعديلات الطفيفة عليها . شكل (6-6) يبيّن دائرة مجمع وقد تم عليها هذا التعديل لتنفيذه عمليات الجمع أو الطرح عن

طريق بعض خطوط التحكم التي يمكن بواسطتها اختيار إما عملية الجمع أو عملية الطرح .

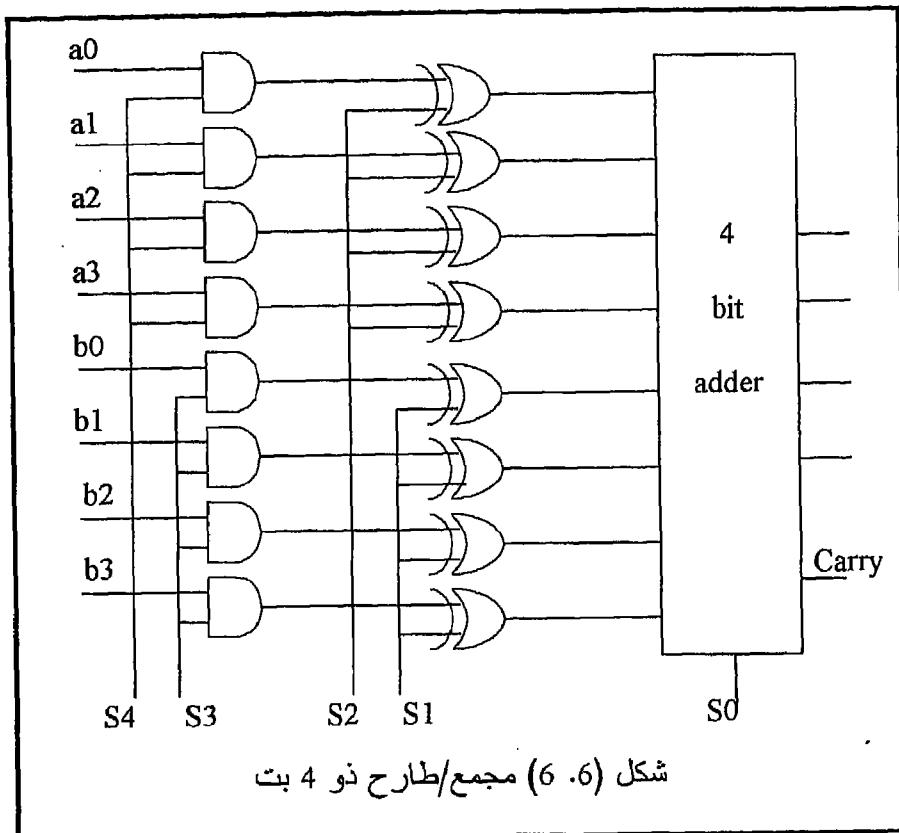


شكل (6.5) التركيب والرمز العام لمجمع ذو 4 بิตات

كما نعلم فإنه من خواص بوابة XOR أن لها دخلان عندما يكون أحدهما يسلوي واحدا فإن الخرج يكون معكوس الدخل الآخر ، وعندما يكون أحد هذين الدخليين يساوى صفرًا فإن الخرج يكون مثل الدخل الآخر ، أي أن بوابة XOR يمكن استخدام أحد دخليها للتحكم في جعل الخرج يساوى أما الدخل الآخر أو معكوسه. يمكن أن نرى ذلك بوضوح في شكل (6-6) بحيث أنه إذا كان خط التحكم $S1=1$ فإن العدد B سيدخل للمجمع معكوس . أما إذا كان $S1=0$ فإن العدد B سيدخل إلى المجمع بقيمتها الحقيقة . نفس الشيء تم تطبيقه على العدد A باستخدام خط التحكم $S2$.

من شكل (6-6) أيضا يمكننا استنتاج وظيفة خطى التحكم $S4, S3, S2, S1$ بأن كل منها بمثابة مفتاح (ON/OFF) للعدد الذي يعمل معه . فالخط $S3$ سيسمح بمرور العدد B أو صفر بدلا منه ، والخط $S4$ سيسمح بمرور العدد A أو صفر بدلا منه . إنه باستخدام خطوط التحكم $S4, S3, S2, S1$ يمكن استخدام المجمع الموجود في شكل (6-6) في أكثر من وظيفة منها الجمع والطرح . إفترض مثلا أن $S4=1$ ،

في هذه الحالة بما أن $S_1=0$, $S_0=0$, $S_3=1$, $S_2=0$, واحداً فإن العددان A, B سيعبران من بوابات ال AND بنفس قيمهما وبما أن كل منها يساوى صفرًا فإن العددان A, B سيعبران من بوابات ال XOR بنفس قيمهما الحقيقة ومن ذلك نرى أن المجمع سيكون دخله المباشران هما العددان A, B وصفر من خط التحكم S_0 لذلك سيقوم المجمع بجمعها.



شكل (6.6) مجمع/طراح ذو 4 بت

افترض الآن الوضع التالي $S_4=1$, $S_3=1$, $S_2=0$, $S_1=1$, $S_0=1$ من شكل (6-6) نرى أنه طالما أن S_4 , S_3 كل منها يساوى واحداً فإن العددان A, B سيعبران من بوابات ال AND ، وبما أن $S_2=0$ فإن العدد A سيعبر من بوابات ال XOR بقيمه الحقيقة وطالما أن $S_1=1$ فإن العدد B سيعبر من بوابات ال XOR معكوساً وبذلك نستطيع القول أن المجمع سيقوم بعملية الجمع التالية :

$$A + \overline{B} + 1$$

حيث الواحد هو قيمة خط التحكم S_0 . كما علمنا من قبل فإن $A + B = \overline{B} + A$. لذلك فإننا نستطيع القول بأن المجمع في هذه الحالة يقوم بعملية طرح العدد B من العدد A ، أي $(A - B)$.

شكل (6-6) به خمسة خطوط تحكم هي S_4, S_3, S_2, S_1, S_0 كل منها يمكن أن يكون واحداً أو صفراء لذلك فإن هناك 32⁽²⁾ شفرة أو كود يمكن أن تكون عليها هذه الخطوط ولكل شفرة من هذه الشفرات سيكون هناك خرج معين لدائرة المجمع/الطارح الموضحة في شكل (6-7) . شكل (6-7) يبين جميع هذه الشفرات والخرج الناتج عن كل منها . إننا هنا لن نقوم بمراجعة جميع الحالات الموجودة في شكل (6-7) لمعرفة كيف يكون الخرج في كل حالة ومتباقة ذلك على الدائرة الموجودة في شكل (6-6) ولكننا سنترك هذه العملية للقارئ اكتفاء بالمثالين الذين شرحناهما سابقاً أحدهما للجمع والأخر للطرح . من الملاحظات المهمة على الخرج أن هناك الكثير من الحالات قد يكون فيها الخرج غير مهم مثل الحالات التي يكون فيها الخرج يساوى 0 أو 1 أو 2 . وغير ذلك من الحالات ، ويلاحظ أيضاً وجود الكثير من الحالات التي يكون الخرج فيها مكرراً مثل الحالة التي يكون فيها الخرج يساوى B قد كررت ثلاثة مرات والحالة التي يكون فيها الخرج يساوى 0 كررت أيضاً أكثر من مرة ، كيف سنتخلص من هذا التكرار وهذه الحالات التي تعتبرها غير مهمة ؟ إن هذا ما سنراه في الأجزاء القادمة .

6-4 وحدة الحساب والمنطق

Arithmatic and Logic Unit, ALU

من العمليات التي يقوم بها المعالج دائماً بجانب العمليات الحسابية العمليات المنطقية أيضاً . لذلك فإننا سنحاول في هذا الجزء وعن طريق إضافة بعض خطوط التحكم أن نجعل الدائرة الموجودة في شكل (6-6) قادرة على تنفيذ العمليات المنطقية أيضاً بجانب العمليات الحسابية (الجمع والطرح) . إن العمليات المنطقية التي سنحاول إضافتها إلى وحدة المجمع/الطارح التي سبق شرحها هي عمليات XOR, OR, AND . وهذا كمثال فقط حيث بالطبع يمكن إضافة المزيد .

إن هناك أكثر من طريقة لتطوير دائرة المجمع/الطارح التي سبق شرحها لتسهيل تنفيذ العمليات المنطقية وسنعرض هنا أبسط هذه الطرق . شكل (6-8) يبين وحدة حساب ومنطق تستطيع تنفيذ عمليات الجمع والطرح و XOR و OR و AND . في هذا الشكل أن الدخلين A, B كل منهما مكون من n من البتات ولكن لتبسيط الرسم تم رسمه خط واحد فقط . شكل (6-8) مكون من أربعة صناديق كل منها يمثل عملية من عمليات وحدة الحساب والمنطق . الصندوق

الأول خاص بعمليتي الجمع والطرح ومحاتوياته هى الدائرة الموجودة فى شكل . (6-6)

s4	s3	s2	s1	s0	الخرج
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	-1
0	0	0	1	1	0
0	0	1	0	0	-1
0	0	1	0	1	0
0	0	1	1	0	-2
0	0	1	1	1	-1
0	1	0	0	0	B
0	1	0	0	1	B+1
0	1	0	1	0	-B-1=B
0	1	0	1	1	-B
0	1	1	0	0	B-1
0	1	1	0	1	B
0	1	1	1	0	-B-2
0	1	1	1	1	-B-1=B
1	0	0	0	0	A
1	0	0	0	1	A+1
1	0	0	1	0	A-1
1	0	0	1	1	A
1	0	1	0	0	-A-1=A
1	0	1	0	1	-A
1	0	1	1	0	-A-2
1	0	1	1	1	-A-1=A
1	1	0	0	0	A+B
1	1	0	0	1	A+B+1
1	1	0	1	0	A-B-1
1	1	0	1	1	A-B
1	1	1	0	0	B-A-1
1	1	1	0	1	B-A
1	1	1	1	0	-A-B-2
1	1	1	1	1	-A-B-1

شكل (6-7) جميع الحالات الممكنة لخرج المجمع/الطارح فى شكل (6-6)

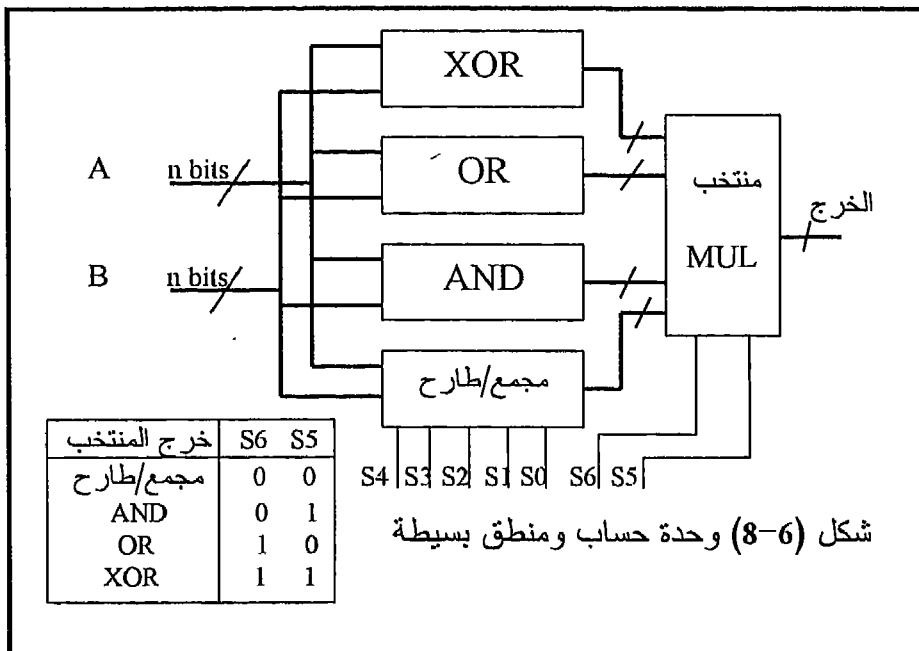
الصندوق الثاني خاص بعملية AND ويحتوى عددا n من بوابات AND ثنائية الدخل حيث أحد هذين الدخلين هو بت من بتات الدخل A والدخل الآخر هو بت المراقبة من الدخل B . الصندوق الثالث خاص بعملية OR ومحتوياته هي عدد n من بوابات OR ثنائية المدخل ، وأما الصندوق الرابع فخاص بعملية XOR ومحتوياته هي عدد n من بوابات XOR .

الخرج النهائي لوحدة الحساب والمنطق يتم اختياره من بين خروج الأربعة صناديق السابقة عن طريق منتخب multiplexer له خطى تحكم S6, S5 تتم بهما عملية اختيار أى واحد من الصناديق سيتم توصيل خرجه إلى خرج المنتخب وبالتالي إلى خرج وحدة الحساب والمنطق ، فمثلا إذا كان $S5=1, S6=1$ فإن خرج صندوق XOR يوصل إلى خرج المنتخب وفي هذه الحالة فإن وحدة الحساب والمنطق ستقوم بتنفيذ عملية XOR على الدخلين A, B . شكل (6-8) يبين أيضا جدول الحقيقة لهذا المنتخب لجميع حالات الخطين S6, S5 . لاحظ أن عملية تخصيص أى شفرة على الخطين S5, S6 ستخرج XOR وأيها ستخرج AND وغير ذلك من العمليات الموضحة في شكل (6-8) تتوقف على المصمم إذ هو الذى يحدد هذه الشفرات . عندما يكون الخطان 0, S5=0 فإن خرج وحدة الحساب والمنطق سيكون هو خرج الصندوق المجمع/الطارح وأما العملية التى ستتعدد فستكون إما عملية جمع أو طرح أو غير ذلك على حسب الشفرة الموجودة على باقى خطوط التحكم S0 إلى S4 .

الدائرة الموجودة في شكل (6-8) تحتوى على سبعة خطوط تحكم S0 إلى S6 وعلى ذلك فإن هذه الدائرة مفروض أن تكون قادرة على إجراء 128 (2⁷) عملية مختلفة بناء على جدول حقيقة يمكن وضعه مماثلا للجدول المبين في شكل (6-7) . ولكن وكما رأينا في شكل (6-7) فإن معظم هذه العمليات (128) إما أنها ستكون عمليات غير مهمة أى غير ذات فائدة أو أنها ستكون عمليات مكررة . لذلك فإننا سنقوم بتصنيفية هذه 128 عملية إلى 13 عملية فقط من العمليات المهمة وغير مكررة وسنهمل باقى العمليات . لاحظ أن اختيار 13 عملية تعتبر من عمل المصمم حيث هو الذى يختار عدد العمليات المهمة وأى العمليات تهم؟ وأيها يؤخذ فى الإعتبار؟ ولذلك فإن اختيارنا 13 عملية هنا ليس إلا مجرد مثال فقط ولا توجد أى ضرورة لاختيار الرقم 13 بالذات .

بما أننا سنختار 13 عملية فقط وسنهمل الباقى فإنه من البديهي أن يكون هناك 4 خطوط تحكم فقط كافية لتشغيل هذه العمليات حيث $4^4 = 16$ (أكبر من 13) . لذلك فإننا سنحتاج هنا إلى دائرة تكون مهمتها هي تحويل الشفرات الرباعية إلى شفرات سباعية تتناسب مع خطوط التحكم الموجودة في وحدة الحساب والمنطق المبينة في شكل (6-8) ، هذه الدائرة سنسميها محول شفرات وعادة ما تكون هذه الدائرة عبارة عن ROM مكونة من 16 بآيت ولها أربعة خطوط عناوين تعطى

عليها الشفرة الرباعية فتخرج محتويات البايت المقابلة وهي الشفرة السباعية التي من المفروض أن تكون مخزنة سلفاً .

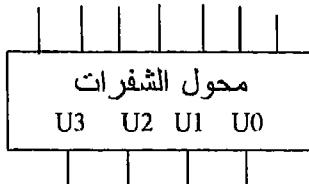


شكل (6-9) يبين رسمياً صندوقياً وجدول الحقيقة لمحول الشفرات . شكل (6-10) يبين الشكل النهائي لوحدة الحساب والمنطق بعد إضافة محول الشفرات إليها . لإجراء عملية الجمع مثلاً يجب أن نضع الشفرة الرباعية 0111 على خطوط التحكم U0 U1 U2 U3 فيقوم محول الشفرات بإعطاء الشفرة السباعية المنشورة والتي هي الشفرة السباعية لعملية الجمع (0011000) كما في جدول الحقيقة في شكل (9-6) .

5- مسجل التراكم Accumulator register

إن إضافة مسجل التراكم Accumulator إلى وحدة الحساب والمنطق هو الخطوة التالية لتطوير هذه الوحدة وإعدادها لتكون جزءاً من أجزاء المعالج . شكل (6-11) يوضح هذا التطوير ، من هذا الشكل نلاحظ أن مسجل التراكم يعتبر مخزناً لتسجيل الناتج من وحدة الحساب والمنطق حيث أنه موصل مباشر على خرجها ولذلك فإننا نتوقع أن ناتج أي عملية حسابية أو منطقية سيذهب مباشرة إلى مسجل

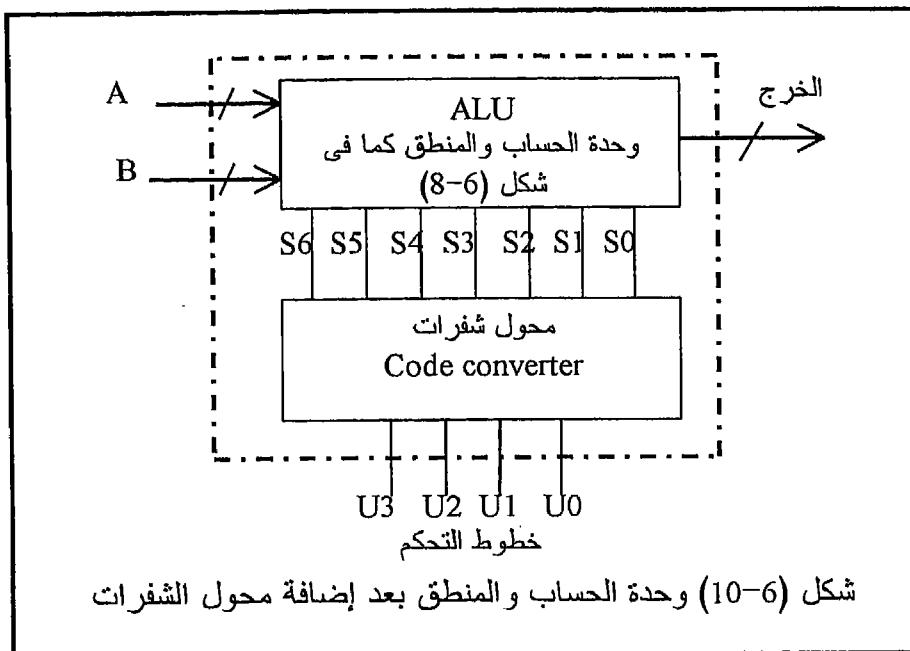
التراكم . نلاحظ أيضا من شكل (6-11) أن هناك نوعا من أنواع التغذية المرتدة حيث قد تم توصيل خرج مسجل التراكم على أحد دخلي وحدة الحساب والمنطق (الدخل A) ولم يبق سوى دخل واحد فقط لوحدة الحساب والمنطق (الدخل B) . نلاحظ أيضا أن مسجل التراكم مثله مثل أي مسجل حيث لن ينتقل دخله إلى خرجه إلا بعد إعطاء نبضة تزامن clock ، لذلك كان لابد من توصيل إلى clock إلى مسجل التراكم .



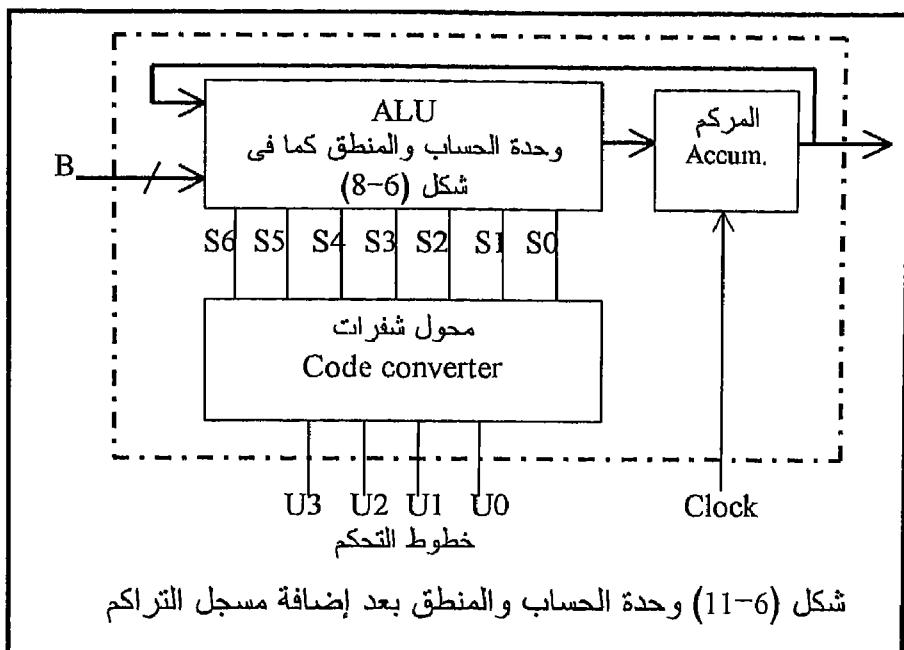
شكل (6-9أ) رسم صندوقى لمحول الشفرات

U_{32}	U_{11}	U_0	العملية	s_6	s_5	s_4	s_3	s_2	s_1	s_0
0	0	0	A	0	0	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0	1
0	0	1	0	A	0	0	1	0	1	0
0	0	1	1	B	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	0	1	A+1	0	0	1	0	0	1
0	1	1	0	A-1	0	0	1	0	0	1
0	1	1	1	A+B	0	0	1	1	0	0
1	0	0	0	A-B	0	0	1	1	0	1
1	0	0	1	A \wedge B	1	0	d	d	d	d
1	0	1	0	AVB	0	1	d	d	d	d
1	0	1	1	A+B	1	1	d	d	d	d
1	1	0	0	-1	0	0	0	0	1	1
هذه الحالات الثلاث										
1	1	0	1	غير مستخدمة الآن						
1	1	1	0	وسوف نستخدمها مستقبلا						
* d تعنى أي لا يهم أن يكون هذا الخط صفر أو واحد .										

شكل (6-9ب) جدول الحقيقة لمحول الشفرات



بعد هذا التعديل بإضافة مسجل التراكم إلى وحدة الحساب والمنطق لابد وأن يطرأ تعديل مناظر على الـ 13 عملية التي تم تخصيصها لهذه الوحدة للقيام بها والتي سبق أن حددها سابقاً بعد إضافة محول الشفرات في شكل (6-9). شكل (6-12) يبين جدولًا لهذه العمليات وقد تم إعطاء شفرة حرفية mnemonic أو شفرة أسمبلى لكل عملية بجانب شفرتها الثنائية (الشفرة الرباعية) أو التي سنسميها شفرات الماكينة machine codes . لذا نأخذ على ذلك المثال التالي : افترض أن خطوط التحكم U3 U2 U1 U0 تساوى 0101 فإنه من شكل (6-9) ستكون العملية (الأمر) التي ستتفذ هي $A+1$ ، أي أن الدخل A لوحدة الحساب والمنطق الذي هو خرج مسجل التراكم سيزداد بمقدار واحد . بافتراض أن خرج مسجل التراكم كان صفرًا في البداية فإن الـ ALU ستجمع واحد زائد صفر وتكون النتيجة واحداً يوضع على خرج الـ ALU . مع أول نبضة تزامن فإن هذا الواحد ينتقل إلى خرج مسجل التراكم حيث سيكون موجوداً أيضاً على الدخل A للـ ALU . ومع بقاء نفس الشفرة على الخطوط U0 إلى U3 فإن هذا الواحد سيصبح اثنين على خرج الـ ALU . بإعطاء نبضة التزامن الثانية ستنتقل الاثنين إلى خرج مسجل التراكم والدخل A للـ ALU وهكذا تستمرة العملية . أي أن هذا الأمر يزيد واحداً على محتويات مسجل التراكم مع كل نبضة تزامن كما لو كان عدداً ولذلك فقد تم اختصاره إلى INC أي زد بمقدار واحد Increment .



شكل (11-6) وحدة الحساب والمنطق بعد إضافة مسجل التراكم

كما نلاحظ من شكل (6-11) فإن خرج مسجل التراكم متصل دائماً بالدخل A لوحدة الحساب والمنطق لذلك فإنه دائماً يطلق عليه اسم المسجل A ، وبما أن النتائج دائماً تترافق فيه كما رأينا في المثال السابق فقد أطلق عليه أيضاً اسم مسجل التراكم أو المركم Accumulator . لنأخذ مثلاً آخر أكثر تعقيداً لكي نفهم كيفية عمل الدائرة الموجودة في شكل (11-6) . افترض أن المطلوب هو ضرب العددين 3×6 . كما نرى من شكل (6-12) فإنه ليس هناك أى أمر يقوم بعمليّة الضرب ، لذلك فإننا سننفذ الضرب عن طريق الجمع المتكرر ، أى أننا سنجمع العدد 6 مع نفسه ثلاثة مرات . لذلك فإننا سنضع العدد 6 على الدخل B أولاً ثم سنضع الشفرة 0011 على خطوط التحكم U0 إلى U3 وهذه الشفرة من شأنها أن تنقل الموجود على الدخل B إلى خرج مسجل التراكم مع أول نبضة تزامن ، أى أنه بعد أول نبضة تزامن سيكون العدد 6 موجوداً على خرج مسجل التراكم وبالتالي أيضاً على الدخل الآخر (الدخل A) لوحدة الحساب والمنطق . الآن سنضع الشفرة 0111 على خطوط التحكم U0 إلى U3 وهذه الشفرة كما في شكل (6-12) ستقوم بجمع محتويات الدخل B مع محتويات الدخل A لـ ALU . وعلى ذلك فإنه بإعطاء نبضة تزامن ثانية ستكون محتويات مسجل التراكم تساوى 12 وهي حاصل جمع 6 الموجودة على الدخل B و 6 الموجودة على خرج مسجل التراكم بعد النبضة الأولى . لاحظ أن الدخل A لـ ALU بعد

النبضة الثانية سيصبح 12 أيضا . لو احتفظنا بنفس الشفرة 0111 على خطوط التحكم U0 إلى U3 وأعطيينا نبضة تزامن أخرى فإن خرج مسجل التراكم سيصبح 18 وهي حاصل جمع 6 الموجودة على الدخل B و 12 الموجودة على خرج مسجل التراكم بعد النبضة الثانية . بذلك تكون قد أنهينا عملية ضرب العدددين 6×3 بعد ثلات نبضات تزامن وبعد أن وضعنا الشفرات التالية على خطوط التحكم U0 إلى U3 :

	U3	U2	U1	U0
أول نبضة تزامن	0	0	1	1
ثاني نبضة تزامن	0	1	1	1
ثالث نبضة تزامن	0	1	1	1

من المثال السابق يمكننا أن نقول أن أي عملية مركبة يمكن تبسيطها إلى مجموعة من الشفرات التي توضع على خطوط التحكم U0 إلى U3 واحدة بعد الأخرى بحيث تتفذ العملية المناظرة لكل شفرة مع نبضة تزامن . الآن لا يمكننا أن نسمى كل شفرة من هذه الشفرات بالأمر Instruction ، ومجموعة الشفرات أو الأوامر المطلوبة لتنفيذ أي عملية مركبة إلا نسميها برنامج program ، إلا يمكن أن نسمى الدائرة التي لها قائمة الأوامر الموجودة في شكل (6-12) بالمعالج . نعم إن الدائرة التي حصلنا عليها في شكل (11-6) تعتبر صورة مبسطة جدا لمعالج افتراضي محدود الإمكانيات إذا ما قورن بأى معالج حقيقي . إن المعالج الإفتراضي hypothetical الذى وصلنا إليه حتى الآن يستطيع فقط إجراء العمليات الحسابية والمنطقية ، السؤال الآن هل نستطيع تطوير الوحدة السابقة لكي تستطيع التعامل مع ذاكرة ووحدات إدخال أو إخراج كما في المعالج العادى ؟ إن هذا ما سنراه في الجزء التالي :

6-6 إضافة ذاكرة للمعالج الإفتراضي

بالنظر إلى الدائرة الموجودة في شكل (6-13) نجد أن هناك منتخبا تمت إضافته في الدخل وهذا المنتخب له دخلان وخط تحكم واحد S8 تمت إضافته على خرج محول الشفرات لهذا الغرض . أحد دخلي هذا المنتخب هو خرج الذاكرة والدخل الآخر هو الدخل B حيث يتم اختيار أحدهما ليعبر إلى خرج المنتخب ومنه كدخل إلى وحدة الحساب والمنطق ثم إلى مسجل التراكم عن طريق خط التحكم S8 . وعلى ذلك فعندما يكون $S8=0$ فإن خرج الذاكرة يكون موصلا إلى وحدة الحساب والمنطق ، وأما إذا كان $S8=1$ فإن الدخل B يوصل إلى وحدة الحساب والمنطق .

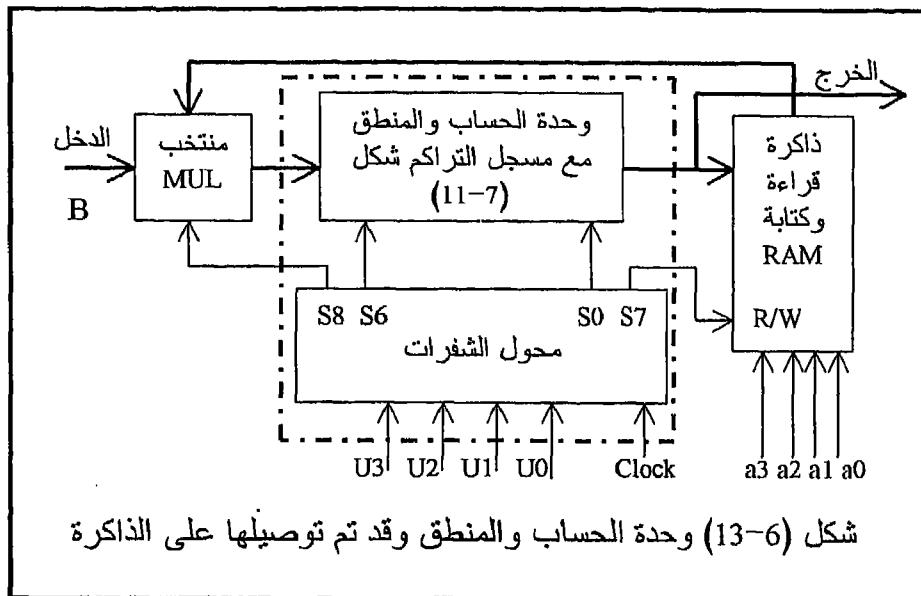
العملية ال اختصار	العملية	الحمل Carry
U3 U2 U1 U0		
0 0 0 0	NOP	لاتعمل شيء
0 0 0 1	SP1	A = +1
0 0 1 0	CMA	اعكس
0 0 1 1	LDA	حمل A بالدخل
0 1 0 0	CLA	A = 0
0 1 0 1	INC	A+1 → A
0 1 1 0	DEC	A-1 → A
0 1 1 1	ADD	A+B → A
1 0 0 0	SUB	A-B → B
1 0 0 1	AND	A&B → A
1 0 1 0	OR	AVB → A
1 0 1 1	XOR	A⊕B → A
1 1 0 0	SM1	-1 → A
1 1 0 1		
1 1 1 0		
1 1 1 1		

* A مقصود بها مسجل التراكم و B يقصد بها الدخل B لوحدة الحساب والمنطق

شكل (6-12) قائمة بالعمليات التي تنفذها وحدة الحساب والمنطق التي معها مرکم كما في شكل (11-6)

المشكلة الآن أننا نريد توصيل أحد هذين الدخلين إلى مسجل التراكم فـأى الشفرات نضعها على خطوط التحكم U0 إلى U3 ؟ ليس هناك إلا شفرة واحدة فقط وهي الشفرة 0011 التي تتبع دخل وحدة الحساب والمنطق في مسجل التراكم وهي شفرة الأمر LDA وكما رأينا فإن دخل وحدة الحساب والمنطق إما أن يكون من الذاكرة أو من B . لذلك فإنه لتمييز كل من الدخلين من الآخر فإننا سنخصص الشفرة 0011 التي هي الأمر LDA لتحميل المرکم بمحتويات دخل وحدة الحساب والمنطق إذا كان هذا الدخل قادماً من الذاكرة أي $S8=0$. ولذلك فإنه في هذه الحالة ($S8=0$) لابد من تحديد العنوان الذي سيتم التعامل معه على خطوط عنوان الذاكرة a0 إلى a3 . أما إذا كان الدخل قادماً من B فإننا سنستخدم له شفرة من الشفرات الغير مستخدمة والمتبعة من الـ 16 شفرة الموجودة في قائمة الأوامر ولتكن الشفرة 1101 والتي سنعطيها الاختصار INP أي الدخل من

خارج المعالج مثل بوابة إدخال مثلاً . لاحظ أنه مع الأمر INP فإن خط التحكم S8 يكون واحداً في هذه الحالة وستكون الشفرة المقابلة على خرج محول الشفرات S0 إلى S8 هي 00001000 .



شكل (6-13) وحدة الحساب والمنطق وقد تم توصيلها على الذاكرة

بذلك تكون قد رأينا كيفية اختيار دخل وحدة الحساب والمنطق بين الذاكرة أو الدخل B القادم من خارج المعالج ، ماذا عن خرج وحدة الحساب والمنطق وكيف يمكن توصيله إلى الذاكرة ؟

إن خط التحكم S7 الذي أضيف على خرج محول الشفرات كما في شكل (6-13) مهمته هي التحكم في كتابة خرج مسجل التراكم في الذاكرة إذا كان $S7=1$ أو إخراجه إلى خارج المعالج عندما يكون $S7=0$. لاحظ أن كلاً من هاتين العمليتين ليس لها أمر يقابلها في قائمة الأوامر التي درسناها حتى الآن في شكل (6-12) ولذلك فإننا سنستخدم الشفتين المتبقيتين من الـ 16 شففة في نفس الشكل لهذا الغرض . أي أن الشففة 1110 والتي سنختصرها STA أي خزن في الذاكرة Store accumulator S7=1 الذي سيجعل الذاكرة في حالة استقبال للمعلومات أي في حالة كتابة فيها لأن الخط R/W=1 . لاحظ أن هذا الأمر سيطلب أن نضع العنوان الذي سيتم التعامل معه داخل الذاكرة على مسار العناوين a0 إلى a3 .

العملية	الامر	العنوان	a3 a2 a1 a0	U3 U2 U1 U0
لاتعمل شيء	NOP	x x x x	0 0 0 0	
$A = +1$	SP1	x x x x	0 0 0 1	
اعكس A	CMA	x x x x	0 0 1 0	
الذاكرة $\leftarrow A$	LDA	a a a a	0 0 1 1	
$A = 0$	CLA	x x x x	0 1 0 0	
$A+1 \rightarrow A$	INC	x x x x	0 1 0 1	
$A-1 \rightarrow A$	DEC	x x x x	0 1 1 0	
عنوان + A	ADD	a a a a	0 1 1 1	
عنوان - A	SUB	a a a a	1 0 0 0	
عنوان $\wedge A$	AND	a a a a	1 0 0 1	
عنوان OR	OR	a a a a	1 0 1 0	
عنوان $\oplus A$	XOR	a a a a	1 0 1 1	
$-1 \rightarrow A$	SM1	x x x x	1 1 0 0	
$B \rightarrow A$	INP	x x x x	1 1 0 1	
$\leftarrow A$	STA	a a a a	1 1 1 0	
$\leftarrow A$ الخرج	OUT		1 1 1 1	

شكل (6-14) قائمة الأوامر بعد تعديلها لتلائم عملية الإتصال بالذاكرة

أما الشفرة 1111 المتبقية فسوف نخصصها لإخراج محتويات المركب خارج المعالج ولذلك سنرمز لها بالرمز OUT أى إخراج output حيث سيكون الخط S7=0 في هذه الحالة . شكل (6-14) يبين قائمة الأوامر بعد تعديلها لتناسب عملية التوصيل للذاكرة وإضافة الأوامر الجديدة إليها .

إننا بعد دراسة هذا الفصل يجب أن نقف وقفة تفكير فى الفرق بين هذا المعالج الإقراصى وأى واحد من المعالج الحقيقى الذى درسناه فى الفصول السابقة . أسلطة كثيرة يمكن أن ترد هنا بعد دراسة الأشكال المختلفة والمتعددة التى رأيناها فى هذا الفصل . كيف سيكون شكل المعالج لو أننا رسمناه على أساس 8 بتات أو 16 أو حتى 32 بت ؟ وماذا لو جعلنا الشفرة الداخلة إلى محول الشفرات شفرة من 16 بت بدلا من الشفرة الرباعية ؟ وكيف سيكون شكل هذه الدائرة إذا أضفنا بعض العمليات الأخرى مثل عمليات دوران محتويات المسجل A إلى اليمين أو إلى اليسار وعمليات الإزاحة من اليمين أو اليسار وعمليات الضرب والقسمة ؟ كيف سيكون شكل هذه الدائرة لو أضفنا لها هذه الوحدة بعض المسجلات عامة للأغراض التى تستخدم لأغراض التخزين وتبادل المعلومات مع المسجل A ؟

كيف سيكون شكل الدائرة لو أن الوحدة تتعامل مع ذاكرة لها 16 أو 32 خطا من خطوط العنوانين بدلاً من 4 كما افترضنا؟ كل هذه أسئلة توضح مدى بساطة فكرة عمل المعالج ولكنها في نفس الوقت توضح مدى تعقيد تركيبه ومكوناته.

7- تمارين

1. دائرة نصف المجمع الموجودة في شكل (6-2) يمكن بناؤها باستخدام بوابات XOR ، وضح ذلك؟
2. دائرة المجمع الكامل الموجودة في شكل (6-4) يمكن بناؤها باستخدام نصف مجمع ، وضح ذلك؟
3. هناك بعض الشرائح التي تعمل كمجمع ، اذكر واحدة من هذه الشرائح وانشرح طريقة عملها واختبارها عمليا؟ استعن بكتاب data book دليل الشرائح.
4. يلعب المتنم الثنائي دوراً مهماً في تحويل عملية الطرح إلى عملية جمع ، وضح ذلك بالشرح؟
5. ما هو دور وحدة الحساب والمنطق في المعالج؟
6. هناك بعض الشرائح التي تعمل كوحدة حساب ومنطق ، اذكر واحدة من هذه الشرائح ، وانشرح طريقة عملها واختبارها عمليا؟
7. شكل (6-9) يبيّن محول شفرات رباعية إلى سباعية ، ماذا يحدث لو استخدمنا محول شفرات خماسية إلى سباعية بدلاً منه؟ اكتب قائمة العمليات الجديدة بعد العمليات الإضافية؟
8. هل مسجل التراكم الموجود في شكل (6-11) يقوم بنفس دور مسجل التراكم في أي معالج من حيث أنه يكون طرفاً في أي عملية حسابية أو منطقية والنتيجة تخزن فيه؟
9. ما مقدار الذاكرة التي يستطيع المعالج الافتراضي الموجود في شكل (6-13) أن يتعامل معها؟ وكيف تزيد هذه الكمية؟
10. لو أردنا إضافة مسجل أوامر للوحدة الموجودة في شكل (6-13) فأين يكون ، وضح ذلك؟

الفصل السابع

أساسيات مواجهة المعالج

*Fundamentals of Microprocessor
Interfacing*

7-1 مقدمة

مهما كان أساسيات تستطيع عملها باستخدام المعالج : المهمة الأولى هي أنك تستطيع برمجته لحل أي مشكلة إذا كان كل ما تطلبه هذه المشكلة هو البرمجة فقط (software) . المهمة الثانية التي تستطيع عملها باستخدام المعالج هي أنك تستطيع مواجهته مع بعض الدوائر الخارجية للوصول إلى الكثير من الأغراض والتي سنذكر منها اثنين فقط على سبيل المثال لا الحصر :

1. تستطيع مواجهة المعالج مع بعض الدوائر الخارجية مثل دوائر الذاكرة وبوابات الإدخال وبوابات الإخراج لعمل حاسب شخصي للأغراض العامة مثل الحاسوبات التي نراها كثيراً في حياتنا اليومية والتي ما منها حاسب إلا قائم أساساً على معالج معين .

2. إنك تستطيع مواجهة المعالج مع بعض الشرائح الخارجية مثل شرائح الذاكرة وبوابات الإدخال والإخراج للحصول على نظام تحكم في عملية صناعية معينة . لكي يتم التحكم في أي عملية صناعية باستخدام المعالج (كالتحكم في سرعة محرك مثلاً) ليس من الضروري أن نضع بجوار هذه العملية كومبيوتر متكون بالحظ الثمين لإتمام عملية التحكم ولكننا نستطيع بناء دائرة مبسطة على كارت واحد مكونة من المعالج وعدد قليل من الشرائح الموصولة عليه لاستيفاء هذا الغرض وباقل التكاليف الممكنة .

إننا في هذا الفصل وقبل أن ندخل في تفاصيل عملية مواجهة أي معالج مع الدوائر الخارجية سنحاول أولاً التعرف على بعض الأساسيات الضرورية والتي يجب أن نأخذها في الاعتبار قبل البدء في عملية المواجهة مع أي معالج .

7-2 فصل خطوط المعالج

Buffering of microprocessor lines

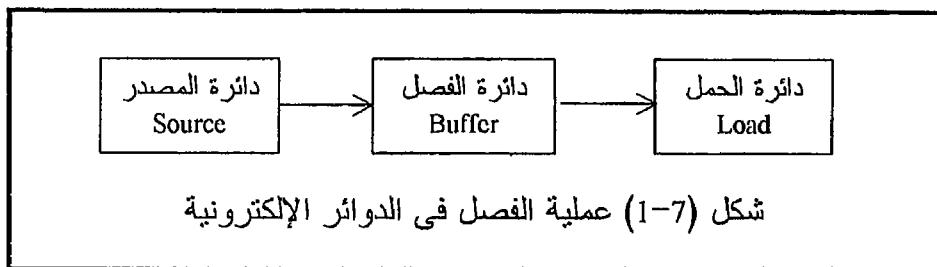
7-2-1 ماذا يعني بكلمة فصل ؟

إن الفصل يكون عادة بين شيئين ينتج من اتصالهما المباشر بعض المشاكل ، لذلك فإننا نضع بينهما وسيطاً يكون حلقة الوصل بين هذين الشيئين . وأقرب مثال على ذلك ما نسمعه في عصرنا الحالي المليء بالحروب عن وجود منطقة فاصلة بين القوتين المتحاربتين تتمرکز فيها قوات محايدة تكون حلقة الوصل بين القوتين . هذا الموقف ينشأ في الكثير من الدوائر الإلكترونية عند تحمل إحداثها على الأخرى . ماذا يحدث لو أن الدائرة المصدر كانت غير قادرة على إدارة driving أو تشغيل الدائرة الحمل بسبب أن الدائرة الحمل تحتاج إلى الكثير من التيار الذي لا تستطيع الدائرة المصدر توفيره ؟ الذي يحدث هو أن جهد خرج

الدائرة المصدر يضمحل أو يتلاشى وبذلك تصبح الدائرة غير قادرة على إدارة الحمل . في هذه الحالة يكون الحل هو استخدام فاصل Buffer بين الدائريتين ويكون هذا الفاصل عبارة عن دائرة إلكترونية (وليس قوات أمم متحدة) تستطيع الدائرة المصدر إدارتها وتستطيع هي إدارة الدائرة الحمل ، ونؤكد هنا على أن الدائرة الفاصلة تكون دائرة يستطيع المصدر إدارتها وإلا فليس هناك أى معنى لاستخدامها كفاصل لأنها في هذه الحالة ستحتاج إلى فاصل . شكل (7-1) يبين عملية الفصل بين دائريتين باستخدام دائرة فصل .

7-2-2 متى نحتاج لفصل buffering خطوط المعالج ؟

من المعروف وكما سنرى في الفصول القادمة أن جميع خطوط المعالج الخارجية منه توصل على الكثير من الدوائر أو الشرائح الإلكترونية على التوازي ، فمثلا خطوط العنوان توصل على العديد من شرائح الذاكرة سواء RAM أو ROM والعديد من بوابات الإدخال وبوابات الإخراج . جميع هذه الشرائح تعتبر أحمالا بالنسبة للمعالج عليه الوفاء باحتياجاتها من التيار ، فعندما يكون خط العنوان الخارج من المعالج High واحد فإن هذه الشرائح ستسحب تيارا معينا من المعالج لابد وأن يستطيع توفيره ، وعندما يكون خط العنوان الخارج من المعالج low أي صفر فإن هذه الشرائح ستصرف تيارات معينة لابد وأن يكون المعالج قادرًا على صرفها أو بلعها .

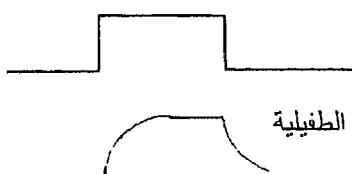


بتجميع هذه التيارات الداخلية والخارجية على الخطوط المتاظرة لجميع الشرائح الموصولة على المعالج يمكننا أن نعرف كم من التيار يجب على شريحة المعالج أن توفرها للشرائح الخارجية في حالة high وكم من التيار يجب أن تستقبلها أو تصرفها sinking في حالة low بعد حساب مجموع التيارات المطلوب توفيرها أو صرفها من المعالج يقوم المصمم بالنظر في الكتالوج الخاص بشريحة المعالج لمعرفة هل يستطيع الوفاء بهذه الأغراض أم لا ؟ بناء على ذلك سيأخذ المصمم قراره بالحاجة إلى فاصل أم لا بناء على المواقف التالية :

- إذا وجد المصمم أن احتياجات الأحمال من التيار ليست أقل مما يستطيع

المعالج توفيره وبكمية كافية كعامل أمان فإنه في هذه الحالة لابد من اللجوء إلى استخدام فاصل buffer ، ويجب على المصمم ألا يضع نفسه في المنطقة الحرجة لأن يهمل استخدام الفاصل إذا وجد أن التيارات التي تحتاجها الأحمال أقل أو تكاد تساوى ما يستطيع المعالج الوفاء به ، بل يجب عليه أن يعطى نفسه عامل أمان كافي لأن ذلك بالطبع سيؤثر على تشغيل الدائرة فيما بعد .

2. إذا كانت المسافة بين الحمل والمعالج طويلة بحيث سيحتاج الوضع لاستخدام أسلاك توصيل cables طولية لإتمام عملية التوصيل ، فإنه في هذه الحالة لابد من استخدام دائرة فصل عند الخروج من المعالج قبل السلك . إن ذلك ناشئ من الطبيعة السعوية لسلك التوصيل ، فمثل هذه الأسلاك تكون لها سعة capacitance وهذه السعة قد تؤثر على شكل الموجة الخارجية من المعالج وتشوهها وبالذات في مثل هذه التطبيقات التي تكون الموجات المربعة هي المستخدمة عادة بحيث تكون ترددات هذه الموجات عالية بحيث تتأثر بمثل هذه السعة الناشئة عن السلك . شكل (7-2) يبين موجة مربعة وقد حصل لها تشويه بسبب السعة الطفيفية parasitic capacitance للسلك . لذلك فإنه عامة نستطيع القول أنه إذا كان الحمل موجوداً على كرت غير الكرت الذي عليه المعالج فإنه في هذه الحالة يستحسن استخدام فوائل buffers على جميع المسارات بعد المعالج مباشرة .



تشويه نتيجة السعة الطفيفية

شكل (7-2) أحد تأثيرات الطبيعة السعوية لأسلاك التوصيل

3. هناك بعض المعالجات التي تستخدم فكرة المزج الزمني time multiplexing بين مساراتها مثل المعالج 8085 الذي يستخدم الثانية خطوط الأولى من مسار العناوين كمسار للبيانات أيضاً بحيث أن هذه الخطوط تحمل إشارة عناوين لمدة معينة من الزمن ثم بعد ذلك تحمل إشارة بيانات كما سنرى ذلك بالتفصيل في الفصل القادم . لمثل هذه المعالجات لابد من إجراء عملية عزل لإشارة البيانات وحدها لتكون على مسار خاص والإشارة العناوين وحدها لتكون على مسار آخر وذلك قبل توصيل الأحمال على هذه المسارات . إذا كان ذلك سيتم فإنه عادة

يستخدم شرائح تقوم بعملية عزل أو فصل للإشارات كل على مسار خاص وفي نفس الوقت تكون هذه الشرائح فوacial buffers تُقى بأغراض الأحمال من التيارات .

الآن إذا تم أخذ القرار بأنه لابد من استخدام الفوacial فهناك بعض الملاحظات التي يجب أن تؤخذ في الاعتبار عند اختيار الشريحة التي ستستخدم كفواacial لأن هناك الكثير من الشرائح التي تستطيع القيام بهذه المهمة :

1. بالطبع كما ذكرنا فإن الفاacial buffer الذي ستستخدمه لابد وأن يكون قادرًا على الوفاء بالتزامات التيار المطلوبة للأحمال وإلا فلا فائدة من استخدامه .

2. لابد أن يكون المعالج يستطيع إدارة جميع الفوacial المركبة على خطوطه وإلا أيضًا فلا فائدة من استخدامها .

3. يجب ألا تؤثر الفوacial على طبيعة الإشارة فهناك مثلاً فوacial من طبيعتها أنها تعكس الإشارة (تجعل الواحد صفرًا والصفر واحدًا) لذلك يجب أن يؤخذ ذلك في الاعتبار إذا كانت مثل هذه الفوacial سوف تستخدم .

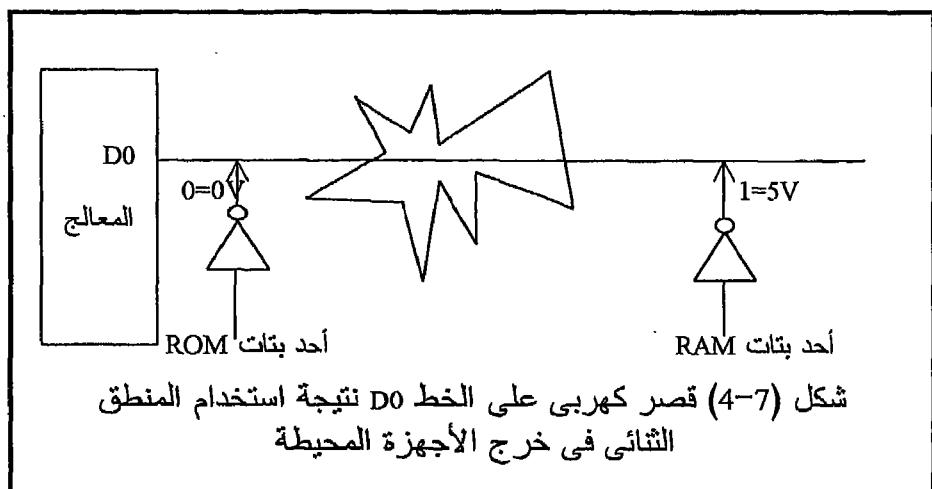
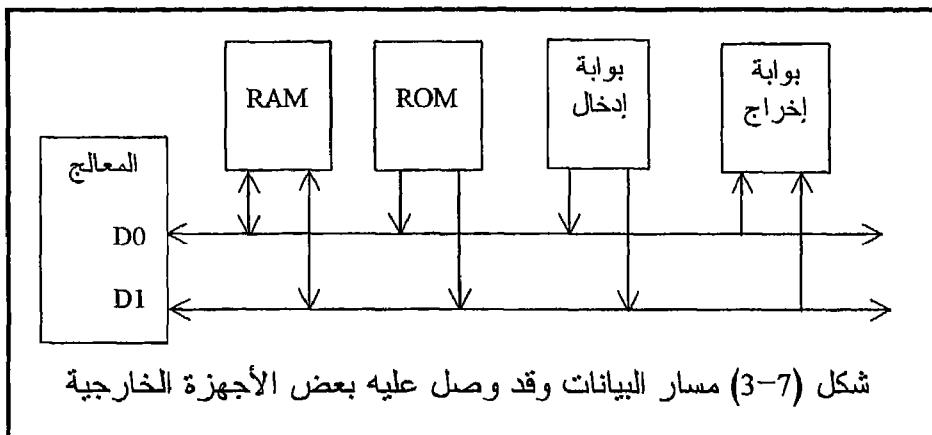
4. يجب أن يناسب الفاacial buffer طبيعة الإشارة التي ستمر من خلاله ، فهناك مثلاً مسارات أحادية الاتجاه بمعنى أن الإشارة عليها تمر في اتجاه واحد فقط مثل مسار العنوانين الذي تكون الإشارة عليه دائمًا من المعالج إلى الأجهزة الخارجية لذلك يجب على الفاacial أن يسمح بذلك ، كذلك فإن من طبيعة مسار البيانات أن الإشارة عليه تمر في كلا الاتجاهين من وإلى المعالج لذلك يجب على الفاacial المستخدم أن يأخذ ذلك في الاعتبار .

7-3 البوابات ثلاثة المنطق Tristate logic gates

إن أي مسار من مسارات المعالج يكون عبارة عن مجموعة من الخطوط المتوازية التي أصلها ، أي تخرج من المعالج ، وموصل عليها على التوازي الكثير من الأجهزة الخارجية مثل شرائح RAM و ROM و بوابات الإدخال والإخراج . من هذه المسارات مسار البيانات الذي يتكون من ثمانية خطوط في بعض شرائح المعالجات التي ندرسها في هذا الكتاب . شكل (7-3) يبيّن هذا المسار خارجاً من المعالج وقد وصل عليه الكثير من الأجهزة المحيطة .

في حالة المنطق الثنائي سنجد أن جميع هذه الأجهزة لابد وأن يكون خرجها إما صفرًا أو واحدًا . لذلك فإنه من الممكن أن يكون الخط D0 مثلاً من مسار البيانات عليه صفرًا من خرج الـ RAM وفي نفس الوقت يكون عليه واحدًا من خرج الـ ROM ، وكما نعلم فإن الواحد المنطقي في نظام الـ TTL يعني جهدًا أكثر من 2.4 فولت والصفر المنطقي يعني جهداً أقل من 0.4 فولت ووجود هذين

الجهدين على نفس الخط وفي نفس الوقت يعني قصر كهربى short circuit مما سينتتج عنه تخريب لمرحلة الخرج فى أحد الجهازين إن لم يكن كليهما . شكل (4-7) يبين خط البيانات D0 وقد حدث عليه هذا القصر short circuit نتيجة توصيل الجهازين اللذين خرجهما عبارة عن مرحلة ثنائية المنطق التى تكون إما صفر أو واحد .

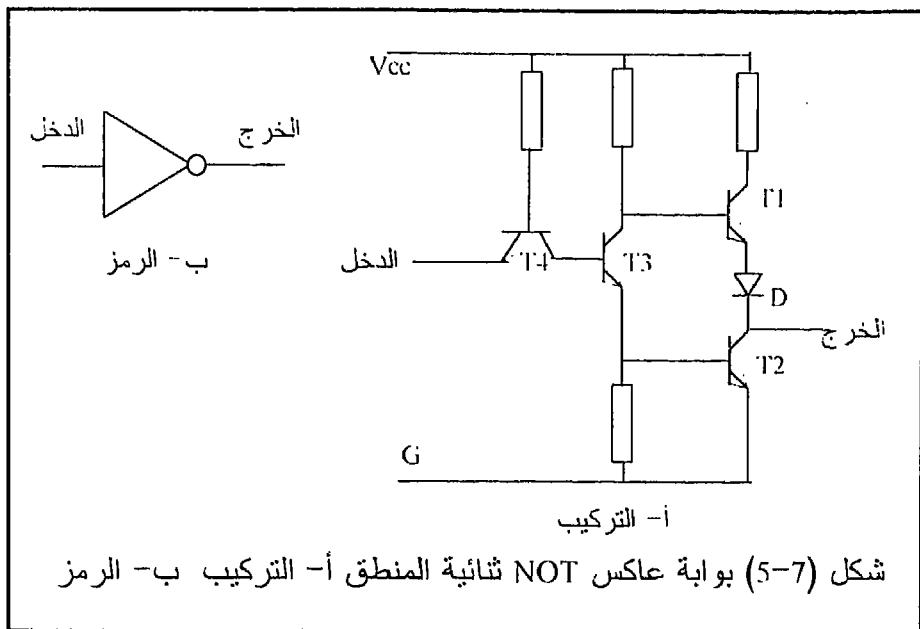


الآن ما هو الحل لهذه المشكلة ؟

إن هذه المشكلة يمكن أن تتلاشى إذا استخدمنا فى مرحلة خرج هذه الأجهزة بوابات ثلاثة المنطق بدلا من البوابات ثنائية المنطق التى نتجت عنها هذه المشكلة . لكي نفهم تركيب البوابات ثلاثة المنطق سنأخذ نظرة سريعة على

تركيب البوابات ثنائية المنطق ولكن قبل ذلك يجب أن نعي جيداً أن هذه المشكلة مصاحبة فقط للأجهزة التي تقوم بادخال معلومات إلى المعالج من خلال مسار البيانات مثل بوابات الإدخال وال RAM وال ROM ، وأما الأجهزة التي تستقبل معلومات من المعالج مثل بوابات الإخراج فلا تعانى من هذه المشكلة .

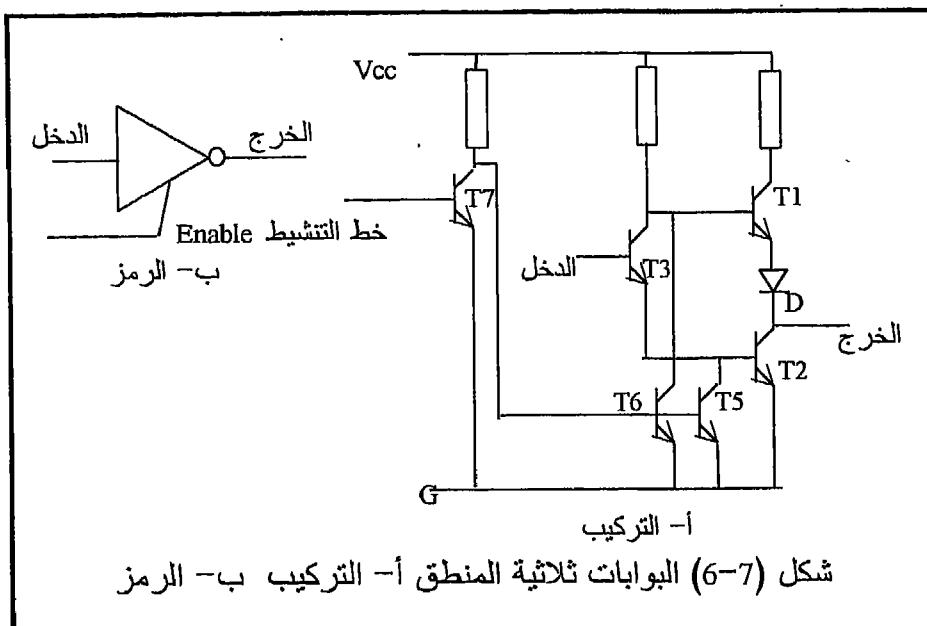
7-3-1 بوابات المنطق الثنائي Double state logic gates



شكل (7-5) بوابة عاكس NOT ثنائية المنطق أ - التركيب ب - الرمز

شكل (7-5) يبيّن تركيب ورمز بوابة من البوابات ثنائية المنطق وهي بوابة NOT . هذه البوابة دائماً يكون خرجها إما واحداً أو صفراء على حسب حالة الدخل . إذا كان الدخل يساوى واحداً H فإن الترانزistor T4 يكون مجمعاً H وبالتالي فإن T3 يكون ON ويكون باعثه H مما يجعل T2 يكون ON وبالتالي فإن الخرج يكون موصلاً إلى الأرضي أي يكون الخرج صفراء في هذه الحالة . لاحظ أن T1 في هذه الحالة يكون OFF أي غير موصلاً ولذلك فإن الخرج يكون معزولاً من مصدر الجهد Vcc . إذا كان الدخل صفراء L فإن مجمعاً T4 يكون ON وبالتالي T3 يكون OFF مما يجعل T2 يكون OFF وبالتالي T1 يكون ON وعلى ذلك فإن الخرج في هذه الحالة يكون متصلًا بمصدر الجهد Vcc ويكون واحداً أي H وفي نفس الوقت ينفصل عن الأرضي .

7-3-2 البوابات ثلاثة المنطق Tristate logic gate

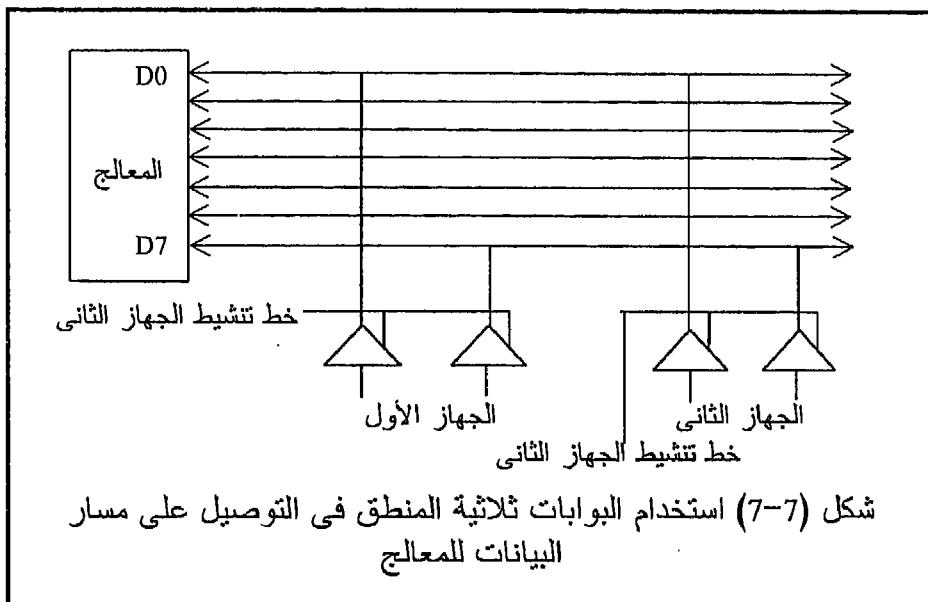


تتميز البوابة ثلاثة المنطق أن لها طرفا ثالثا خاصا بالتحكم في الخرج بحيث إذا كان هذا الطرف فعالا فإن البوابة ثلاثة المنطق تسلك نفس مسلك البوابة ثنائية المنطق تماما ، وأما إذا كان طرف التحكم غير فعال أو خامل فإن خرج البوابة ثلاثة المنطق يأخذ حالة جديدة غير معروفة في البوابات ثنائية المنطق وهي أن الخرج لا يكون صفراء ولا واحدا وإنما يكون مفتوحا open circuit أو مقاومة عالية جدا . high impedance

شكل (7-6) يبين الدائرة الإلكترونية والرمز المستخدم لمرحلة خرج بوابة ثلاثة المنطق وقد تم التحكم فيها عن طريق خط التنشيط enable . لاحظ أن مرحلة الخرج هذه هي نفسها مرحلة الخرج التي تم شرحها في شكل (5-7) ولكن مضاد عليها الترانزسترات T7, T6, T5 التي تعمل كمفاتيح يتم التحكم فيها عن طريق خط التنشيط enable . فإذا كان خط التنشيط عاليا H فإن T7 يكون ON مما يجعل T6, T5 كل منهما يكون OFF وبالتالي فإن البوابة تعمل كبوابة ثنائية المنطق مثل التي شرحت سابقا في شكل (5-7) بحيث إذا كان الدخل H فإن الخرج يكون L والعكس . أما إذا كان خط التنشيط خاملا L فإن T7 يكون OFF مما سيجعل T6, T5 كل منهما يكون ON وبالتالي فإن كلا من T2, T1 يكون OFF وبالتالي فإن الخرج يكون غير موصل لا على الأرضي ولا على الجهد Vcc ولكن يكون كما لو كان مفتوحا open circuit أو مقاومة عالية.

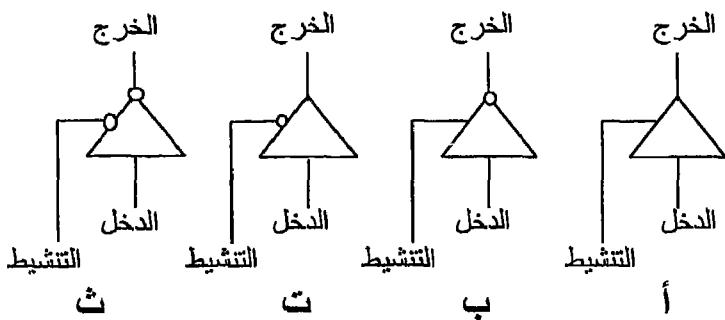
السؤال الآن كيف ستستخدم البوابات ثلاثة المنطق في الحماية من التصر الكهربائي short circuits الذى يحدث بسبب توصيل أكثر من جهاز على نفس خطوط المسارات كما أوضحتنا فى شكل (4-7) ؟

إن جميع الأجهزة التى ستوصى على مسار البيانات للمعالج يجب أن تكون مرحلة الخرج فيها عبارة عن بوابات ثلاثة المنطق وعن طريق خطوط التنشيط لكل جهاز فإن المعالج سيجعل جميع الأجهزة فى حالة خمول أى أن خرجها سيكون كما لو كان غير موصى على المدار إلا جهازا واحدا فقط وهو الجهاز الذى يتعامل معه المعالج فى تلك اللحظة . أى أنه نتيجة استخدام هذا النوع من البوابات فلن يكون هناك غير جهاز واحد فقط هو الفعال فى أى لحظة وهو الذى يتعامل معه المعالج وهو الذى سيكون موصلا على مسار البيانات وأما بقية الأجهزة فستكون منفصلة عن مسار البيانات نتيجة أن خط التنشيط الخاص بها غير فعال . شكل (7-7) يبين عملية توصيل أكثر من جهاز على مسار البيانات باستخدام البوابات ثلاثة المنطق .



شكل (7-7) استخدام البوابات ثلاثة المنطق فى التوصيل على مسار البيانات للمعالج

شكل (7-8) يبين الأنواع المختلفة للبوابات ثلاثة المنطق . هناك مثلا البوابات التى يكون خرجها مثل دخلها تماما إذا كان خط التنشيط فعالا ، كما أن هناك البوابات التى يكون خرجها عكس دخلها إذا كان خط التنشيط فعالا . هناك أيضا البوابات التى يكون خط تنشيطها فعالا عندما يكون صفراء ، وأخرى يكون خط تنشيطها فعالا عندما يكون واحدا . مهمتك أنت كمصمم هى اختيار البوابة المناسبة للتطبيق الذى تستخدمه .



- الخرج = الدخل إذا كان خط التنشيط يساوى واحد
- الخرج عكس الدخل إذا كان خط التنشيط يساوى واحد
- الخرج = الدخل إذا كان خط التنشيط يساوى صفر
- الخرج عكس الدخل إذا كان خط التنشيط يساوى صفر

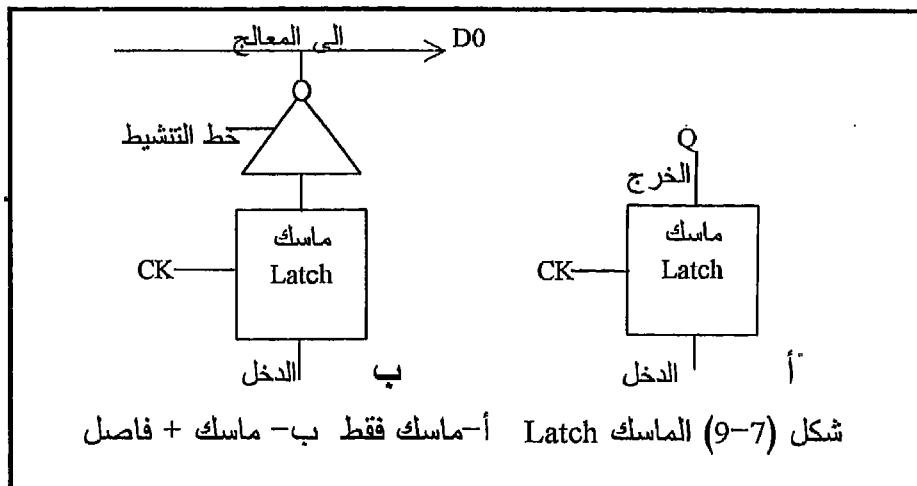
شكل (7-8) الأشكال المختلفة للبوابات ثلاثة المنطق

4-7 الماسك Latch

في بعض التطبيقات تتطلب عملية الفصل استخدام مكونات لها خاصية مسّك المعلومة على خرجها بالرغم من فقدان هذه المعلومة على الدخول . إن البوابات ثلاثة المنطق ليست لها هذه الخاصية لأنها طالما أن خط التنشيط فعال تكون هناك معلومات معينة على الخرج وب مجرد أن يكون خط التنشيط غير فعال فإن المعلومة التي على الخرج تفقد تماما . إن الماسك عبارة عن قلاب flip flop وغالبا ما يكون من النوع D بحيث أن المعلومة التي على طرف الدخل D تنتقل إلى الخرج Q بعد وجود نبضة على طرف التزامن CK . تظل المعلومة الموجودة على الخرج كما هي لا تتغير حتى لو تغير الدخل D طالما أنه لم تعط أي نبضة تزامن أخرى ، لذلك فإننا نقول إن المعلومة قد مسكت على الخرج .

شكل (7-9أ) يبين مثل هذا الماسك . هناك أيضا الكثير من التطبيقات (أجهزة إدخال البيانات إلى المعالج مثلا) كما سنرى والتي تتطلب وجود بوابة ثلاثة المنطق بعد الماسك لتكون بمثابة فاصل بين مسار البيانات وخرج الماسك وذلك لأن الماسك وحده لا يستطيع توصيله على مسار البيانات مباشرة لأن خرجه يأخذ أحد الحالتين فقط إما الصفر أو الواحد أي ثنائية المنطق . ولذلك فإنه عادة يوضع بعد الماسك بوابة ثلاثة المنطق بحيث يوصل خرج الماسك على مسار البيانات فقط عندما يكون خط تنشيط البوابة ثلاثة المنطق فعلا . شكل (7-9ب)

يبين دائرة ماسك متبوعة ببوابة ثلاثة المنطق .



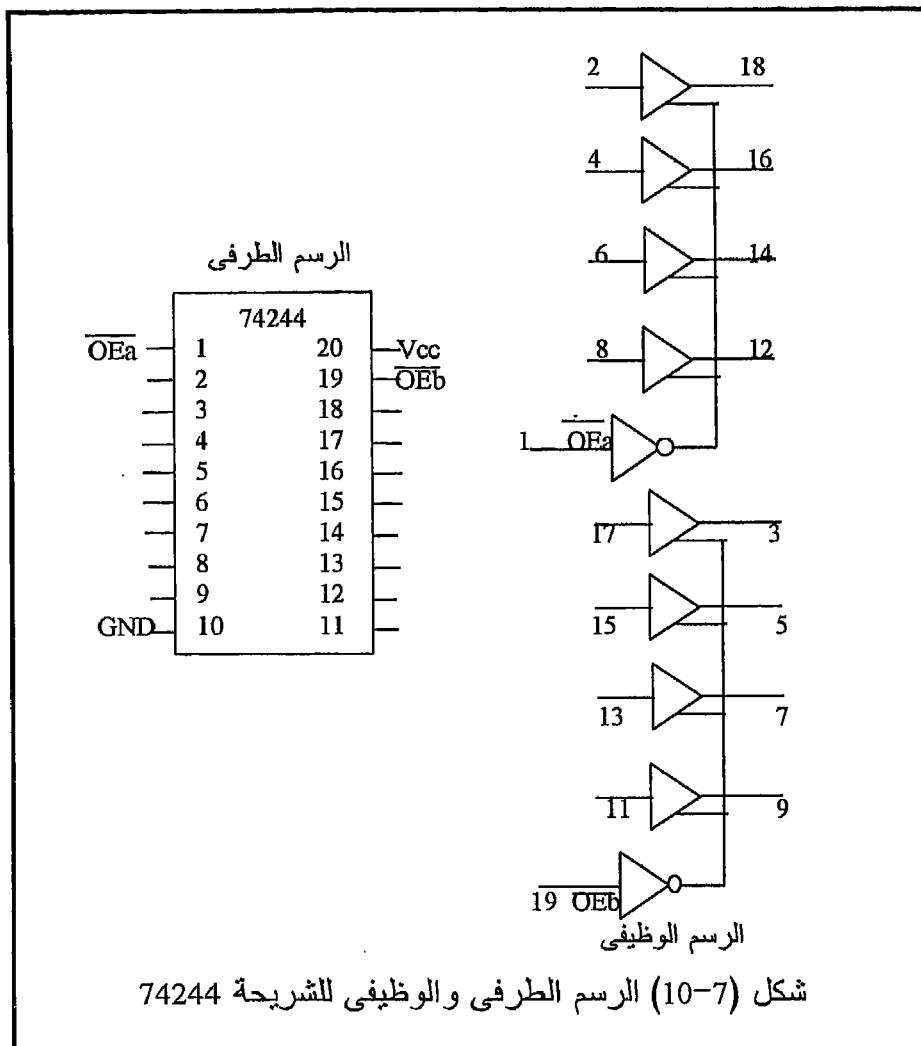
7-5 بعض الشرائح التى تستخدم فى فصل المسارات

سنعرض فى هذا الجزء لمكونات بعض الشرائح التى يمكن استخدامها فى عملية عزل المسارات ويجب أن نؤكد على أنه ليست هذه فقط هى الشرائح المتاحة ولكن هناك الكثير غيرها يمكن استخدامه ، فقط مطلوب قراءة كتالوجات هذه الشرائح وفهمها قبل البدء فى استخدامها .

7-5-1 الشريحة 74244 عازل ثمانى ثلاثة المنطق أحادى الاتجاه

بالنظر إلى الرسم الوظيفي والطرفى للموضحان فى شكل (7-10) يتضح لنا كيفية عمل هذه الشريحة . لاحظ أيضاً أن هذه الشريحة أحادية الاتجاه ، أى أن الإشارة يمكن أن تمر فيها فى اتجاه واحد فقط على عكس الشريحة 74245 التى سيأتي شرحها . هذه الشريحة تحتوى على ثمانى بوابات ثلاثة المنطق مقسمة إلى مجموعتين ، المجموعة الأولى مكونة من 4 بوابات لها خط التنشيط الخاص بها على الطرف رقم 1 والمجموعة الثانية مكونة من الأربع بوابات الأخرى والتي لها خط تنشيط خاص بها هي الأخرى على الطرف رقم 19 . لاحظ أن خطوط التنشيط (1 و 19) تكون فعالة عندما تكون صفراء ، ولذلك فقد تم وضع شرطة فوق اسم كل منها (\overline{OEa} و \overline{OEb}) . كما نعلم عن البوابات ثلاثة المنطق فإنه طالما أن خط التنشيط فعال (صفر فى هذه الحالة) فإن الموجود على دخل

هذه الشريحة ينتقل إلى خرجها . إن هذه الشريحة تصلح فقط لعزل خطوط مسار العناوين وذلك لأن الإشارة التي على هذا المسار تكون دائما خارجة من المعالج إلى الأجهزة المحيطة ولا تأخذ الاتجاه العكسي على الإطلاق .

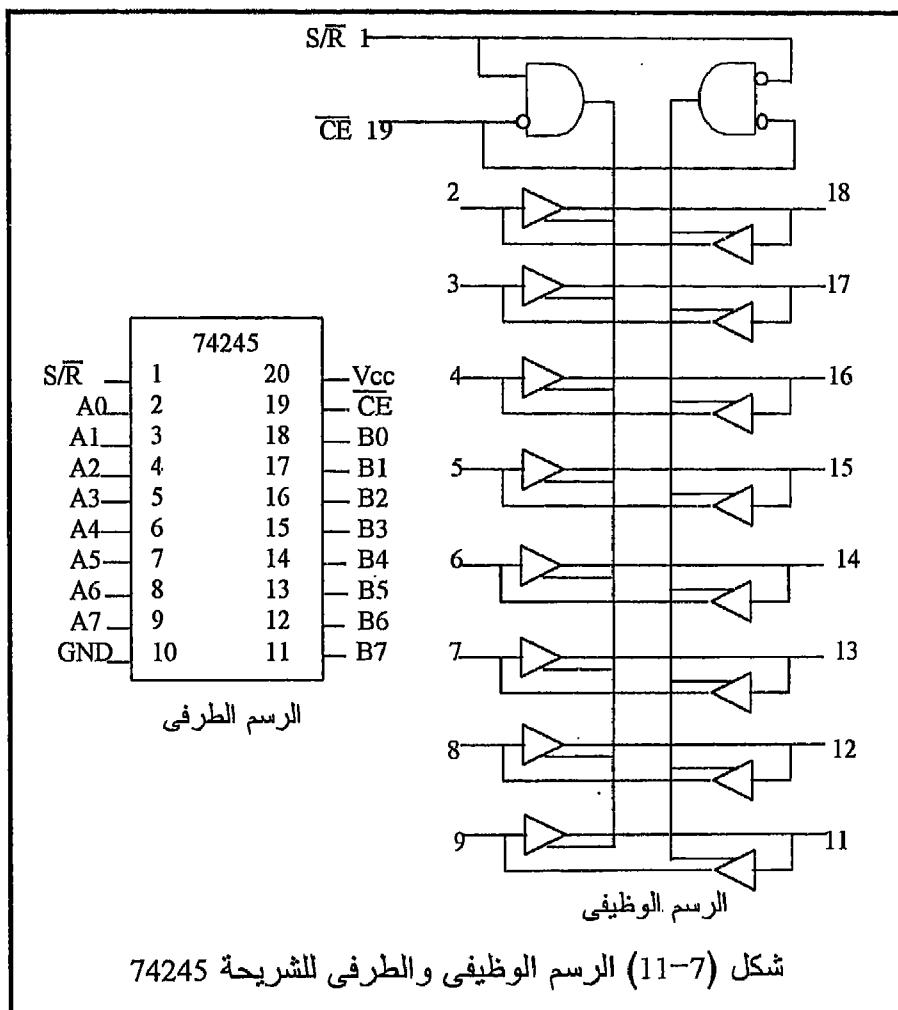


شكل (7-10) الرسم الطرفي والوظيفي للشريحة 74244

5-2 الشريحة 74245 عازل ثماني ، ثلاثي المنطق ، ثانى الاتجاه

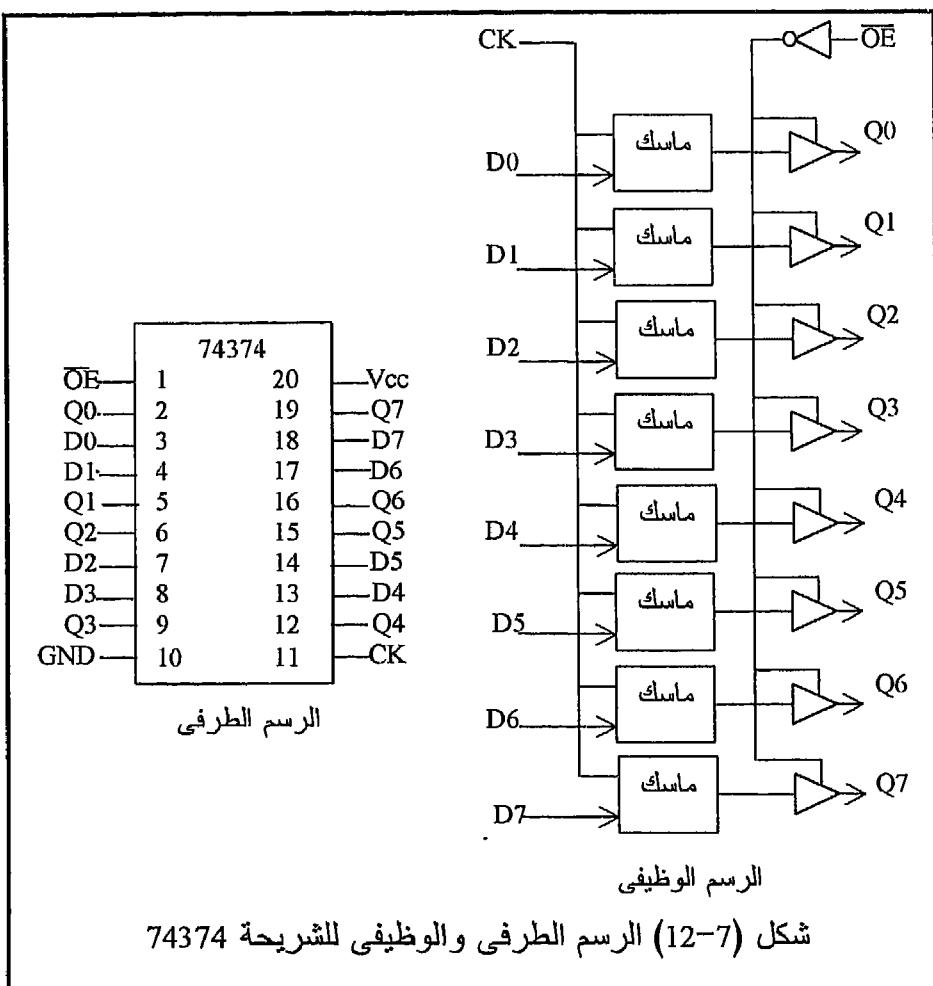
هذه الشريحة عبارة عن عازل buffer ثانى ، الاتجاه أى يمكنها إرسال واستقبال المعلومات ، وخرجها ثلاثي المنطق غير عاكس ولها خط تحكم فى الاتجاه وهو

الطرف رقم 1 . الطرف رقم 19 هو خط تنشيط أو فعالية الشريحة الذى يكون فعال عندما يكون صفر Chip Enable, CE . شكل (7-11) يبين الرسم الطرفى والوظيفى للشريحة . عندما يكون الطرف 19 فعالا (صفر) فإن الشريحة تكون نشطة ، وعندما يكون هذا الطرف خاملا (واحد) فإن الشريحة تكون خاملة ولا تعمل . الطرف 1 يتحكم فى اتجاه البيانات خلال الشريحة . فعندما يكون هذا الطرف واحد فإن الإشارات تمر فى الاتجاه من A إلى B ، وأما إذا كان هذا الخط صفر فإن الإشارة تمر فى الاتجاه من B إلى A . لذلك فإن هذه الشريحة مناسبة جدا لفصل خطوط مسار البيانات كما سنرى .



7-5-3 الشريحة 74374 عازل ماسك ذو ثمانى بتات

تحتوى هذه الشريحة على ثمانية قلابات من النوع D كلها موصولة على نفس طرف التزامن CK وهو الطرف رقم 11 في الشريحة . عند إعطاء نبضة تزامن على هذا الطرف تنتقل الإشارة الموجودة على جميع أطراف الدخل D إلى الخرج المناظر Q ، وتظل هذه الإشارة ممسوكة على الخرج طالما لم يتم إعطاء أي نبضات تزامن أخرى . كل واحد من هذه القلابات موصول على بوابة ثلاثة المنطق تسمح بمرور الإشارة إلى أطراف الخرج عندما يكون طرف التنشيط (الطرف 1) \overline{OE} فعالاً (صفر) . سنرى بعد قليل كيفية استخدام هذه الشريحة في فصل مسار العناوين للمعالج 8085 بالذات لطبيعة مساراته . شكل (7-12) يبين الرسم الطرفي والوظيفي لهذه الشريحة .



شكل (12-7) الرسم الطرفي والوظيفي للشريحة 74374

الفصل الثامن

فصل مسارات المعالجات

Buffering of Microprocessor Buses

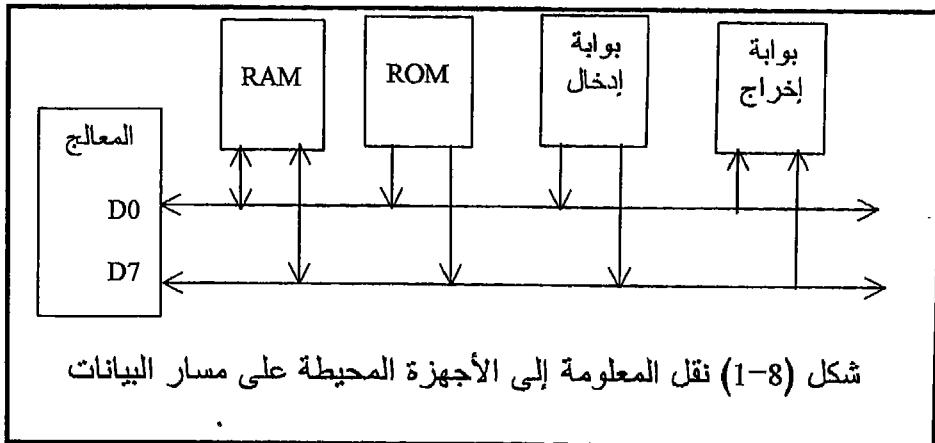
1-8 مقدمة

كما رأينا في الفصل السابق فإن أي مسار bus يكون عبارة عن مجموعة من الخطوط المتوازية والتي عليها يمكن نقل معلومات أو إشارات من مكان لأخر . عادة تكون هذه المعلومات أو البيانات خارجة من مصدر معين وقادمة إلى هدف آخر . بعض هذه المسارات يكون أحدى الاتجاه مثل مسار العنوانين الذي دائماً يحمل إشارات من المعالج إلى الأجهزة المحيطة ، والبعض الآخر يكون ثانية الاتجاه مثل مسار البيانات الذي تكون عليه الإشارة خارجة من المعالج إلى الأجهزة المحيطة في أزمنة معينة أو العكس من الأجهزة المحيطة إلى المعالج في أزمنة أخرى . إن الهدف من عملية مواجهة المعالج مع الأجهزة المحيطة هو توفير الوسائل التي يستطيع بها المعالج التخاطب مع هذه الأجهزة ونعني بكلمة التخاطب إرسال واستقبال معلومات أو إشارات إلى ومن هذه الأجهزة . شكل (8-1) يبين شريحة معالج وقد خرج منها مسار للبيانات إلى جميع الأجهزة المحيطة ، فهل هذا يكفي لحل جميع مشاكل عملية المواجهة ؟ كمثال على الأجهزة المحيطة نرى في هذا الشكل بوابة إدخال وبوابة إخراج وذاكرة قراءة وكتابة RAM وذاكرة قراءة فقط ROM . لاحظ في هذا الشكل أن المعالج كما لو كان بنكاً أو دكاناً للمعلومات وجميع الأجهزة المحيطة تريد التعامل معه من خلال مسار البيانات .

عند مواجهة (توصيل) المعالج مع أي جهاز من الأجهزة المحيطة تنشأ مشكلتان يجب التغلب عليهما وهما كما يلى :

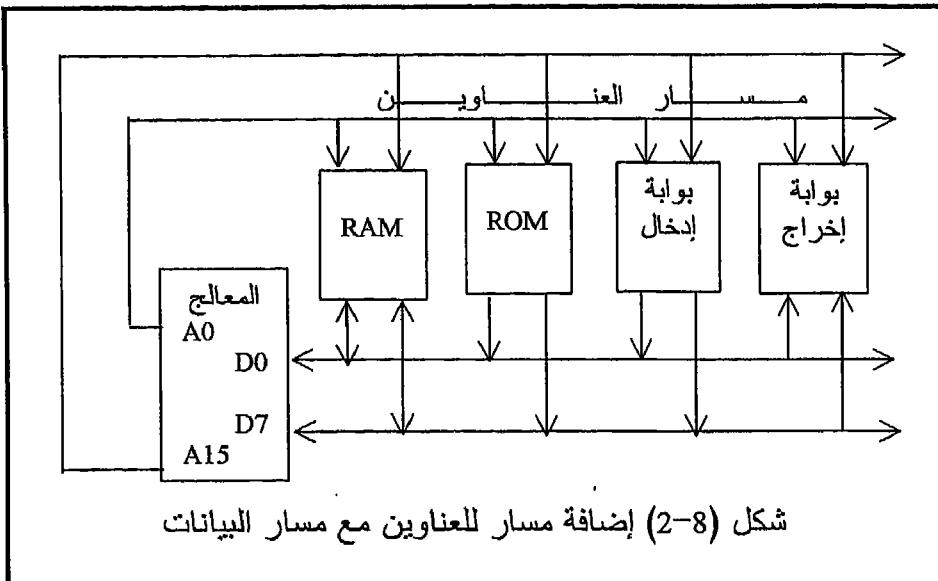
أولاً: يجب التأكد من أنه في أي لحظة لا يتم نقل أي معلومة إلا لجهاز واحد فقط ، أي أنه عندما يكون المعالج في حالة اتصال (مخاطبة) مع أي جهاز من الأجهزة المحيطة فإنه يكون على اتصال بهذا الجهاز فقط دون الأجهزة الأخرى . فمثلاً نريد أن نضمن أنه عندما سيرسل المعالج معلومة إلى أي بوابة إخراج فيلن هذه المعلومة لن تذهب أيضاً إلى أي بait من بايتس الـ RAM .

ثانياً: المشكلة الثانية هي أنه يجب التأكد من أنه عند اتصال المعالج بأى واحد من الأجهزة المحيطة فإن الأجهزة الأخرى لن تشوش أو تتدخل في عملية الاتصال . فمثلاً عندما يريد المعالج أن يقرأ معلومة من الـ RAM فإننا نريد أن نضمن أن الـ ROM أو أي بوابة إدخال لن تتدخل وترسل هي الأخرى معلومات إلى المعالج بحيث يحدث في هذه الحالة ما يسمى بتصادم للمعلومات على مسار البيانات وقد أشرنا لذلك في الفصل السابق في معرض حديثنا عن البوابات ثلاثة المنطق .

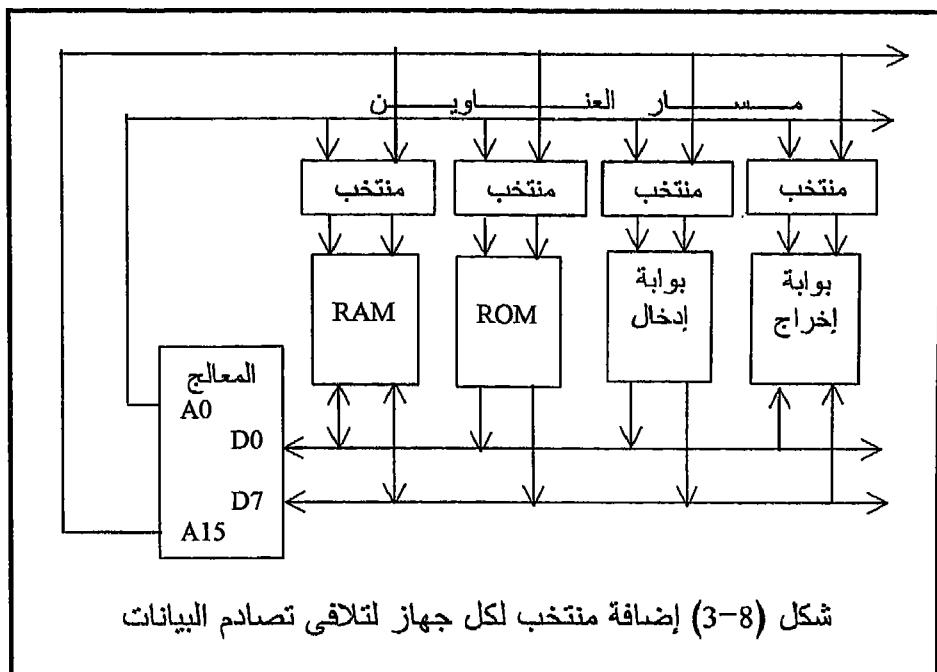


8-2 لماذا مسار العنوان ؟

إن المشكلة الأولى وهي مشكلة ضمان الاتصال أو التعامل مع جهاز واحد فقط يمكن التغلب عليها عن طريق استخدام مسار للعناوين بحيث يكون لكل جهاز من الأجهزة المحيطة عنواناً خاصاً به يتم إخراجه على مسار العنوان من المعالج ولا يتعرف على هذا العنوان إلا الجهاز المعنى به فقط فيصبح هذا الجهاز (الذى تعرف على عنوانه) في حالة نشاط أو فعالية فيستقبل المعلومة الموجودة على مسار البيانات . وأما جميع الأجهزة الأخرى التي لم تتعرف على العنوان فإنها تكون خاملة ويتم منها من التعامل مع مسار البيانات ، لذلك ظهرت الحاجة إلى مسار للعنوانين بجانب مسار البيانات . شكل (8-2) يبين مسار العنوانين وقد أضيف إلى النظام الموجود في شكل (1-8) . عادة يتكون مسار العنوانين من عدد معين من الخطوط ويتم تحديد هذا العدد عن طريق مصمم (صانع) شريحة المعالج . هذا المسار سيحمل إشارة ثنائية (وحابيد وأصفار) وكل إشارة تمثل شفرة أو كودا لعنوان واحد من الأجهزة المحيطة التي يستطيع المعالج التعامل معها . لذلك فإن عدد هذه الأجهزة يساوى اثنين ألس عدد الخطوط الموجودة في مسار العنوانين ، فلو كان مثلاً عدد خطوط مسار العنوانين يساوى ستة خطوط فإن عدد الأجهزة سيكون 64 جهازاً (2⁶) . إن بعض شرائح المعالجات التي سنتعامل معها في هذا الكتاب يحتوى مسار العنوانين فيها على 16 خطأ أو 16 بت ، ولذلك فإنها تستطيع التعامل مع 2¹⁶ أي 65536 جهازاً أو عنواناً من الأجهزة المحيطة بين بوابات إدخال وبوابات إخراج وبآيات ذاكرة كل منها لـ العنوان الخاص به والمكون من 16 بت .



كما رأينا فإن نظام العنونة (كما في شكل (8-2) والذى سيأتي تفصيله فيما بعد) قد حل المشكلة الأولى وهى مشكلة ضمان عدم تعامل المعالج مع أكثر من واحد من الأجهزة المحيطة . أما المشكلة الثانية وهى عدم التداخل بين الأجهزة المحيطة على مسار البيانات أو عدم تصادم المعلومات على نفس المسار فهذه قد أوضحتنا فى الفصل السابق أن سببها يرجع إلى استخدام البوابات ثنائية المنطق فى مراحل خرج الأجهزة المحيطة ، ولقد أوضحتنا فى الفصل السابق أيضاً أن حل هذه المشكلة يكون عن طريق استخدام البوابات ثلاثة المنطق فى مراحل خرج هذه الأجهزة بحيث عندما يريد المعالج التعامل مع أى جهاز فإنه يقوم بتنشيط خط التحكم فى مرحلة خرج هذا الجهاز فقط وأما باقى الأجهزة الموصولة على مسار البيانات ف تكون خاملة أو كما لو كانت غير موصولة على مسار البيانات open circuit . شكل (8-3) يبين عملية توصيل الأجهزة المحيطة مع المعالج وقد استخدم منتخب أو فا لك أو محلل شفرة ملحق بكل جهاز بحيث يصبح خرج هذا المحلل فعالا إذا كان العنوان الموجود على مسار العنوانين مطابقا تماما للعنوان الذى صمم من أجله هذا المحلل ، حيث فى هذه الحالة يكون هذا هو الجهاز الوحيد الذى سيتعامل مع المعالج . إن عملية تصميم منتخب أو محلل شفرة لكل جهاز من الأجهزة المحيطة س تعرف عليها بالتفصيل فى الفصول القادمة فى معرض الحديث عن مواجهة الذاكرة وبوابات الإدخال والإخراج .



شكل (8-3) إضافة منتخب لكل جهاز لتلافي تصادم البيانات

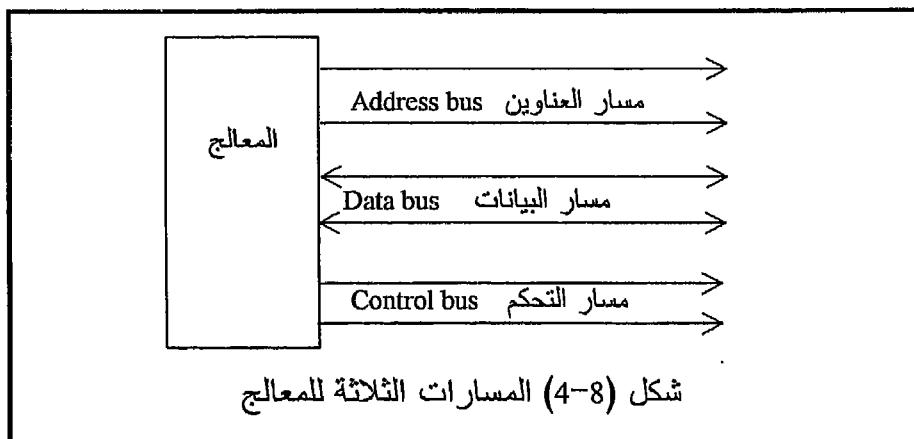
8-3 لماذا مسار التحكم ؟

افرض مثلاً أن المعالج يريد كتابة (إرسال) المعلومة أو الرقم H 55H إلى البايت التي عنوانها E100H ، فماذا سي فعل المعالج على ضوء معرفتنا بوظيفة كل من مسار العناوين والبيانات ؟ إن المعالج لكي يقوم بهذه المهمة فإنه سيضع العنوان E100H على مسار العناوين وبذلك تصبح شريحة الذاكرة التي تحتوى هذه البايت نشطة وعلى استعداد للتعامل مع المعالج ، عند ذلك يقوم المعالج بوضع المعلومة H 55H على مسار البيانات فتتقاها البايت المعنية وتسجل فيها . المشكلة هنا هي أن المعالج عندما قام بتنشيط شريحة الذاكرة التي تحتوى هذه البايت لم يخبر الشريحة بما إذا كان سيرسل إليها معلومات أم سيسقبل منها ، أى هل سيكتب فيها أم سيقرأ منها . لذلك كان من الضروري أن يكون هناك خط تحكم يخرج من المعالج يخبر الجهاز الذي سيعامل معه المعالج عن الهدف من هذا التعامل ، هل هو بعرض القراءة أم بعرض الكتابة . مثل هذا الخط وخطوط أخرى تجمع تحت اسم مسار التحكم control bus وعدد الخطوط في هذا المسار يختلف من معالج لآخر . سنذكر هنا أهم أربعة خطوط تحكم وسنترك الباقي

لكلام عنه في مواضع استخدامه .

من خطوط التحكم ما يلى :

1. خط قراءة الذاكرة memory read, MEMR وهذا الخط يقوم المعالج بتنشيطه في حالة القراءة من الذاكرة (ROM أو RAM) .
2. خط الكتابة في الذاكرة memory write, MEMW وهذا الخط يقوم المعالج بتنشيطه في حالة الكتابة في الذاكرة (RAM) .
3. خط قراءة بوابة إدخال input port read, IOR وهذا الخط يكون فعالاً عندما يكون المعالج في حالة استقبال معلومات من بوابة إدخال .
4. خط كتابة في بوابة إخراج output port write, IOW وهذا الخط يكون فعالاً عندما يكون المعالج في حالة إرسال للمعلومات إلى بوابة إخراج .
لاحظ أن واحد فقط من هذه الخطوط (خطوط التحكم) يكون فعالاً في لحظة وباقى الخطوط تكون خاملة ولذلك فإن تسمية هذه المجموعة من الخطوط بالمسار تعتبر تسمية مجازية ومن الصواب أن تسمى خطوط التحكم فقط ولكن جرى العرف على إطلاق اسم مسار التحكم عليها . شكل (4-8) يبين شريحة معالج وقد خرج منها المسارات الثلاثة : العنوانين والبيانات والتحكم . لاحظ أن عدد خطوط مسار العنوانين 16 خطأ في بعض المعالجات التي ندرسها في هذا الكتاب (وهي المعالجات ذات 8 بت) وسيصل عدد خطوطه إلى 32 بت كما سنرى في المعالجات الحديثة . وكذلك عدد خطوط مسار البيانات 8 خطوط في المعالجات ذات 8 بت وسيصل إلى 32 أيضاً كما سنرى . أما عدد خطوط مسار التحكم فلم يكتب في شكل (4-8) لأن هذا العدد كما أشرنا يختلف من معالج لآخر .



4-4 تهيئة مسارات المعالج 8085 لعملية المواجهة

X1	1	40	Vcc
X2	2	39	HOLD
Reset out	3	38	HLDA
SOD	4	37	CLK OUT
SID	5	36	Reset in
TRAP	6	35	READY
RST7.5	7	34	IO/M
RST6.5	8	33	S1
RST5.5	9	32	<u>RD</u>
INTR	10	31	<u>WR</u>
<u>INTA</u>	11	30	ALE
AD0	12	29	S0
AD1	13	28	A15
AD2	14	27	A14
AD3	15	26	A13
AD4	16	25	A12
AD5	17	24	A11
AD6	18	23	A10
AD7	19	22	A9
Vss	20	21	A8

٨٠٨٥

شكل (5-8) أطراف الشريحة 8085

إذا ألقينا نظرة فاحصة على أطراف الشريحة 8085 كما في شكل (5-8) في محاولة للتعرف على المسارات المختلفة لهذا المعالج لوجدنا الآتي :

- مسار العنوانين يمكن التعرف على 8 خطوط فقط منه وهي الخطوط A8 إلى A15 أما باقى الخطوط فليست موجودة بالصورة المباشرة .
- مسار البيانات أيضا من الصعب التعرف عليه بالصورة المباشرة .
- ماذا تعنى الخطوط AD0 إلى AD7 هل هي خطوط لمسار العنوانين أم لمسار البيانات؟

4. خطوط التحكم كما عرفناها مسبقا ليست موجودة أيضا بالصورة المباشرة ، ولكن الموجود هو الخطوط RD و WR فهل هذه الخطوط لها علاقة بخطوط القراءة من الذاكرة MEMR و الكتابة في الذاكرة MEMW التي تكلمنا عنها في معرض الحديث عن مسار التحكم ؟

جميع هذه الأسئلة وزيادة سنجيب عنها في هذا الجزء في محاولة للحصول على

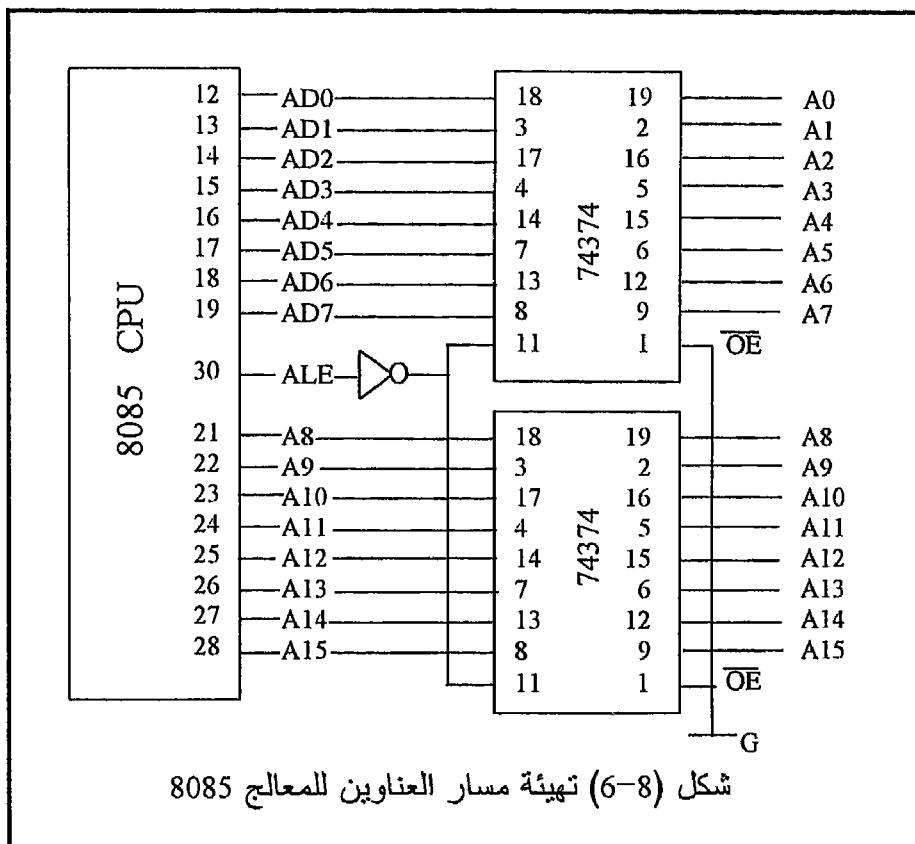
المسارات الثلاثة في الصورة المباشرة والملائمة لعملية توصيل هذا المعالج مع الأجهزة المحيطة .

8085 مسار العنوان للمعالج 1-4

تختلف الشريحة 8085 عن الكثير من الشرائح من حيث أن كل من مسار العنوان والبيانات يستخدم نفس الخطوط AD0 إلى AD7 في عملية مشاركة زمنية time multiplexing بحيث أن الإشارة الموجودة على هذه الخطوط تكون إشارة عنوان في بداية كل دورة أمر ثم تكون بعد ذلك إشارة بيانات . أى أن الإشارة الموجودة على الخطوط AD0 إلى AD7 تمثل عنوان للحظة وجيزة في بداية كل دورة أمر ثم تختفي إشارة العنوان وتتصبح الإشارة الموجودة هي إشارة بيانات ، ولذلك فإننا لو استطعنا مسك إشارة العنوان أثناء هذه اللحظة على ماسك لحصلنا على العنوان بالكامل A0 إلى A15 . السؤال هنا هو : هل هناك وسيلة لمعرفة متى تكون الإشارة على هذه الخطوط الثمانية AD0 إلى AD7 تمثل عنوانين ومتي تمثل بيانات ؟ لقد أجاب البروسيسور 8085 على هذا السؤال وأعطانا الخط ALE على الطرف 30 والذي عن طريقه يمكن معرفة نوع الإشارة على الخطوط AD0-AD7 . إن الحروف ALE تعنى Address Latch Enable أي منشط ماسك العنوان . هذا الخط يكون واحد عندما تكون الإشارة على الخطوط AD0-AD7 تمثل عنوانين ، ويكون صفرًا عندما تكون الإشارة على هذه الخطوط تمثل بيانات . بذلك نستطيع استخدام هذا الخط كخط تحكم أو خط تشغيل لشريحة ماسك تقوم بمسك أو تخزين الإشارة على الخطوط AD0-AD7 عندما يكون الخط ALE يساوى واحداً وبذلك تكون قد حصلنا على العنوان بالكامل A0-A15 . شكل (8-6) يبين الخطوط AD0-AD7 وقد أدخلت على الشريحة 74374 التي هي عبارة عن ماسك ثمانى كما شرحناها فى الفصل السابق . ولقد تم توصيل الطرف ALE من المعالج إلى طرف التزامن clock للشريحة 74374 من خلال عاكس NOT حتى نضمن أن عملية مسح العنوان ستتم عند نزول الخط ALE من الواحد إلى الصفر بناء على طلب المعالج . فى الشكل (8-6) نلاحظ أن الخطوط A8-A15 قد أدخلت هي الأخرى على شريحة ماسك مثل الخطوط AD0-AD7 فهل هناك ضرورة لذلك ؟

فى الحقيقة ليست هناك ضرورة لذلك ولكننا استخدمنا الشريحة 74374 فى هذا المكان لتحقيق عملية فصل buffer لهذه الخطوط حتى نستطيع إمداد جميع الدوائر المحيطة بالتيارات اللازمة . لاحظ أن الطرف رقم 1 (\bar{OE}) فى الشريحة رقم 74374 وهو خط التحكم فى البوابات ثلاثة المنطق الموجودة فى مرحلة خرج هذه الشريحة قد تم توصيله على الأرضى حتى تكون مرحلة الخرج فعالة دائمًا ، أى أنه بمجرد مسح العنوان فإنه يصبح مباشرة موجوداً على خرج

الشريحة . يمكن توصيل هذا الخط بالطرف HOLD القادر من المعالج لوضع مسار العنويين في حالة المقاومة العالية عند اللزوم . شكل (8-7) يبين التزامن الموجود بين الإشارة ALE والإشارات الموجودة على الخطوط AD0-AD7 وكذلك الخطوط AD8-AD15 . لاحظ أهمية تخزين محتويات الخطوط AD0-AD7 عند الحافة الهاابطة لخط التحكم ALE .

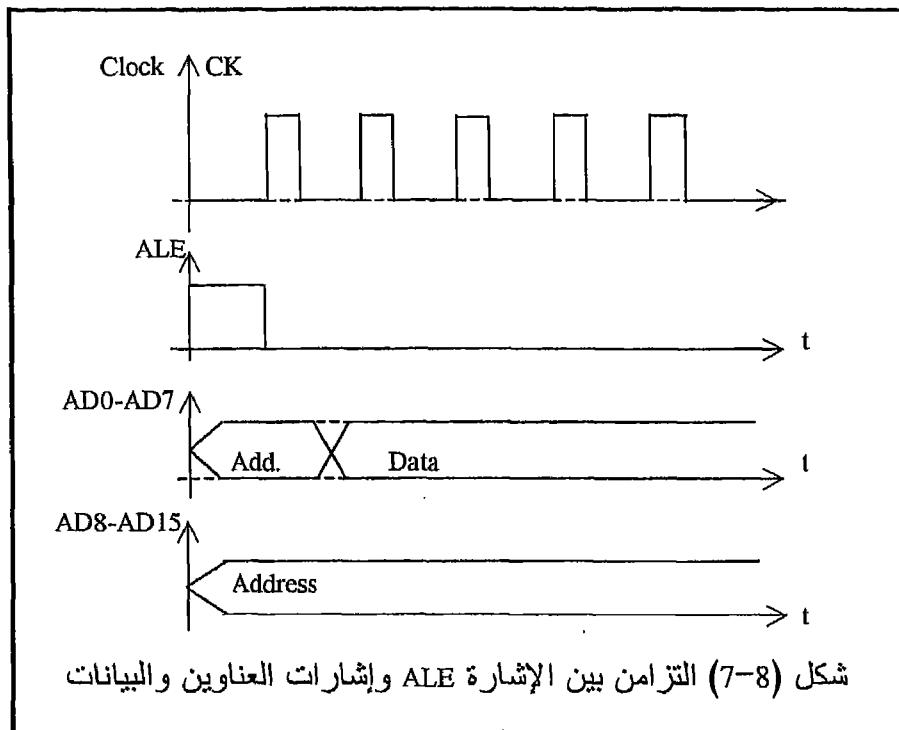


شكل (8-6) تهيئة مسار العنويين للمعالج 8085

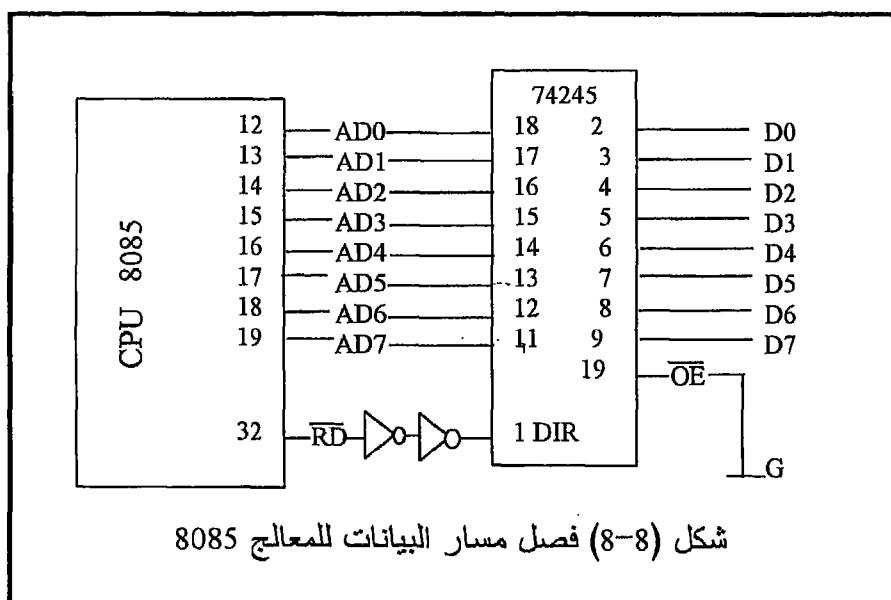
8085-2 مسار البيانات للشريحة

الآن وقد تم فصل مسار العنويين وتهيئته فإن الإشارة الموجودة على الخطوط AD0-AD7 تمثل إشارة بيانات في الزمن المتبقى من دورة الأمر ، والمطلوب فقط هو عملية فصل buffer لمسار البيانات حتى يستطيع توفير التيارات اللازمة للأجهزة المحيطة الموصولة عليه . لاحظ أن مسار البيانات ثنائى الاتجاه لذلك يجب مراعاة استخدام الشريحة المناسبة له وقد أوضحنا في الفصل السابق

أنه من الشرائح المرشحة لهذه العملية الشريحة 74245 والتي سبق شرحها .



شكل (7-8) التزامن بين الإشارة ALE وإشارات العنوانين والبيانات



شكل (8-8) فصل مسار البيانات للمعالج 8085

شكل (8-8) يبين عملية فصل أو تهيئة مسار البيانات . نلاحظ في هذا الشكل أن الطرف رقم 1 في الشريحة 74245 هو طرف التحكم في اتجاه الإشارة \overline{DIR} ولقد تم توصيل هذا الطرف بالطرف رقم 32 في المعالج وهو طرف القراءة \overline{RD} ، بحيث عندما يكون هذا الطرف (\overline{RD}) فعالاً أى يساوى صفرًا فإن الشريحة 74245 ستسمح بمرور البيانات من الأجهزة المحيطة إلى المعالج . بينما إذا كان الطرف \overline{RD} غير فعال ، أى يساوى واحد ، فإن الشريحة 74245 ستسمح بمرور البيانات من المعالج إلى الأجهزة المحيطة . نلاحظ أيضاً من شكل (8-8) أن الخط \overline{RD} قبل توصيله إلى الشريحة 74245 تم فصله عن طريق توصيله من خلال عاكسين .

8-4-3 مسار التحكم للشريحة 8085

إن مسار التحكم البسيط يتكون كما ذكرنا سابقاً من 4 خطوط فقط وهي كالتالي:

- قراءة من الذاكرة Memory read
- كتابة في الذاكرة Memory write
- قراءة من جهاز إدخال Input device read
- كتابة في جهاز إخراج Output device write

طرف 32 \overline{RD}	طرف 31 \overline{WR}	طرف 34 $\overline{IO/M}$	
0	1	1	\overline{IOR}
1	0	1	\overline{IOW}
0	1	0	\overline{MEMR}
1	0	0	\overline{MEMW}

شكل (8-9) جدول حقيقة للحصول على خطوط التحكم الأربع للمعالج 8085

جميع هذه الخطوط فعالة عندما تكون صفرًا active low ولو فحصنا أطراف الشريحة 8085 فإننا لن نجد أن هذه الخطوط الأربع بالصورة المباشرة التي نريدها ولكننا سنجد ثلاثة خطوط فقط وهي الخطوط \overline{RD} و \overline{WR} و $\overline{IO/M}$ و \overline{M} والمطلوب هو الحصول على خطوط التحكم الأربع السابقة من هذه الخطوط الثلاثة . إن السر يكمن في الطرف $\overline{IO/M}$ حيث أن هذا الخط يكون واحداً عندما يكون المعالج يتعامل مع أجهزة إدخال أو إخراج أى ينفذ واحد من الأمرين IN أو OUT ، كما أن الخط $\overline{IO/M}$ يكون صفرًا في حالة ما إذا كان المعالج يتعامل مع ذاكرة . ولذلك إذا كان الخط \overline{RD} فعالاً (1) ، وكان الخط $\overline{IO/M}$ يساوى صفرًا ، فإن ذلك يعني أن المعالج في حالة قراءة من الذاكرة . أما إذا كان الخط

\overline{RD} يساوى صفر ، والخط $\overline{IO/M}$ يساوى واحد ، فإن ذلك يعني أن المعالج فى حالة قراءة من جهاز إدخال . شكل (8.9) يبين جدول الحقيقة لجميع الخطوط الأربع المطلوبة وحالة كل خط من الخطوط الثلاثة \overline{RD} و \overline{WR} و $\overline{IO/M}$. بعد دراسة جدول الحقيقة المبين فى شكل (8.9) يمكن بناء أكثر من دائرة يمكن دخلها هو الخطوط \overline{RD} و \overline{WR} و $\overline{IO/M}$ وخرجها هو الخطوط الأربع $MEMR$ و $MEMW$ و IOW و IOR .

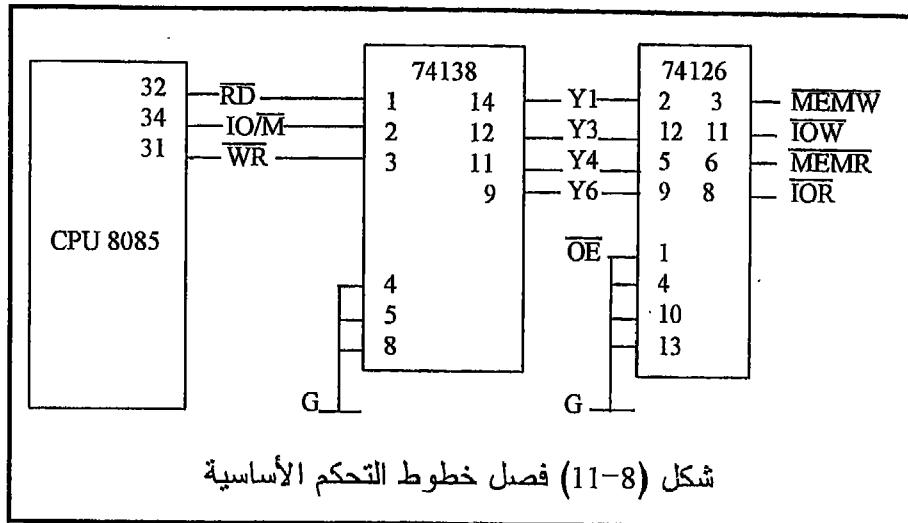
سنرى هنا طريقة الحصول على هذه الخطوط باستخدام المنتخب أو فاڪاك الشفرة Decoder رقم 74138 وهو عبارة عن منتخب لخط واحد من خطوط الخرج الثمانية Y_0 إلى Y_7 وهذا الانتخاب يكون على حسب شفرة توضع على خطوط الدخل الثلاثة A, B, C . شكل (8-10) يبين جدول الحقيقة لهذا المنتخب وقد تم توصيل دخوله الثلاثة على الخطوط \overline{RD} و \overline{WR} و $\overline{IO/M}$.

الدخل INPUT			الخرج OUTPUT							
\overline{WR}	IO/\overline{M}	\overline{RD}		$MEMW$		IOW	$MEMR$		IOR	
C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	H	H	H	H	H	H
L	H	L	H	H	L	H	H	H	H	H
L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	H	H	H	L	H	H	H
H	L	H	H	H	H	H	H	L	H	H
H	H	L	H	H	H	H	H	H	L	H
H	H	H	H	H	H	H	H	H	H	L

شكل (8-10) جدول الحقيقة للشريحة 74138

طبقاً لهذا الشكل فإن الخط $MEMR$ سيؤخذ من على الخرج Y_4 للمنتخب ، والخط $MEMW$ يؤخذ من على الخرج Y_1 ، وأما الخط IOR فيؤخذ من على الخرج Y_6 ، والخط IOW يؤخذ من على الخرج Y_3 . وأما باقى خروج المنتخب فإنها غير مستخدمة . شكل (8.11) يبين كيفية توصيل هذا المنتخب مع المعالج . لاحظ من هذا الشكل أن خطوط الخرج الأربع تم توصيلها على الشريحة 74125 وهى فاصل buffer ثلاثي المنطق وقد وصلت جميع خطوط تشبيط البوابات بالأرضى حتى تكون هذه الخطوط فى حالة نشاط دائم . يمكن عند الضرورة توصيل خطوط التشبيط هذه بالخط HOLD القادم من المعالج .

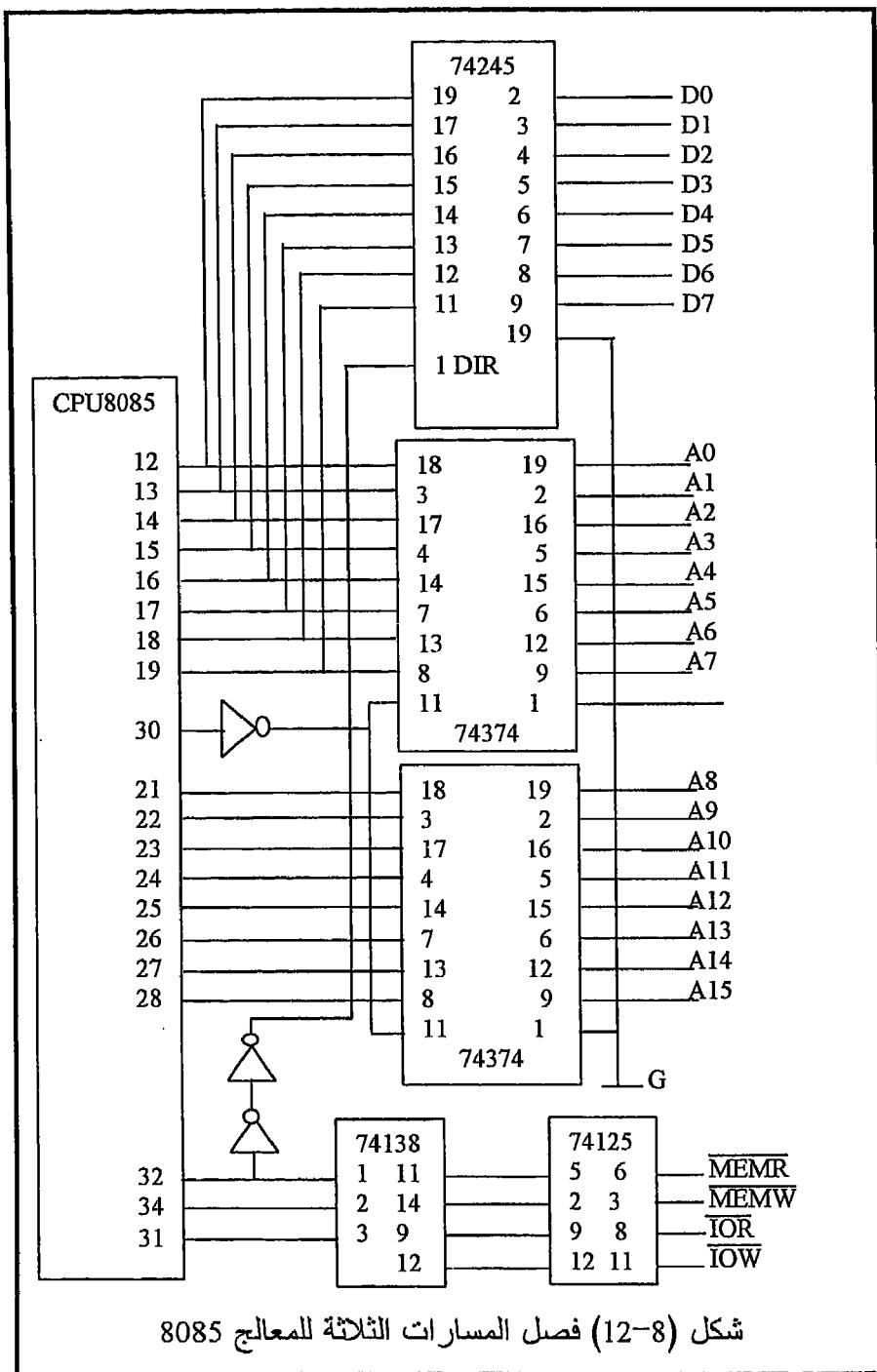
شكل (8-12) يبين المعالج 8085 وقد تم فصل buffering جميع مساراته الثلاثة وأصبحت هذه المسارات مجهزة تماماً لأن يصل إليها أي واحد من الأجهزة الخارجية مثل الذاكرة وبوابات الإدخال والإخراج كما سنرى بالتفصيل في الفصول القادمة .



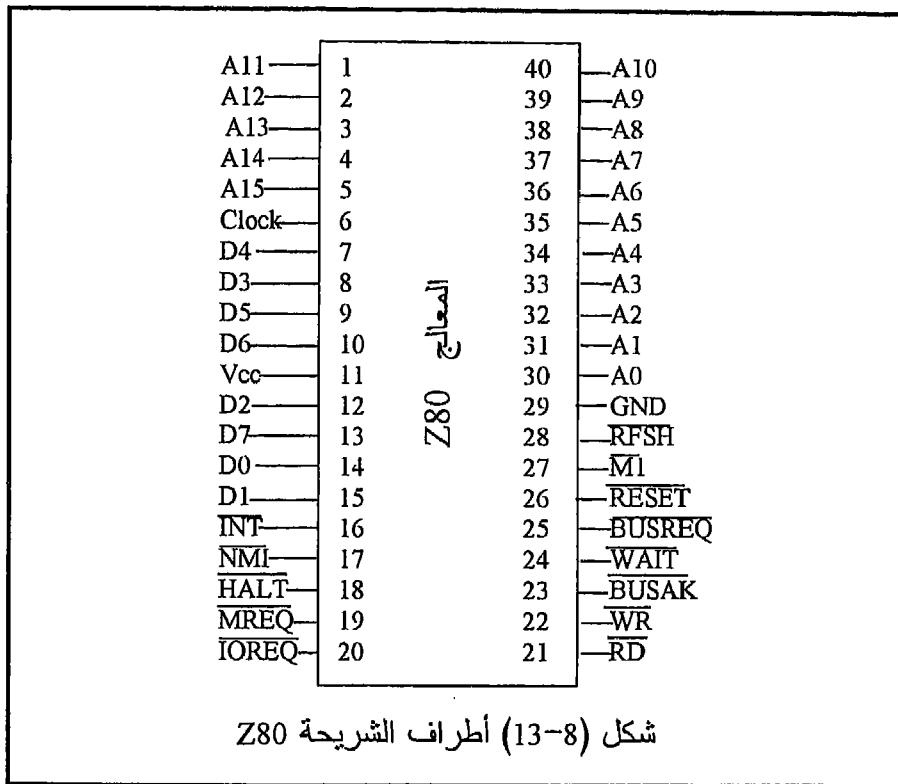
شكل (11-8) فصل خطوط التحكم الأساسية

5-5 تهيئة مسارات المعالج Z80 لعملية المواجهة

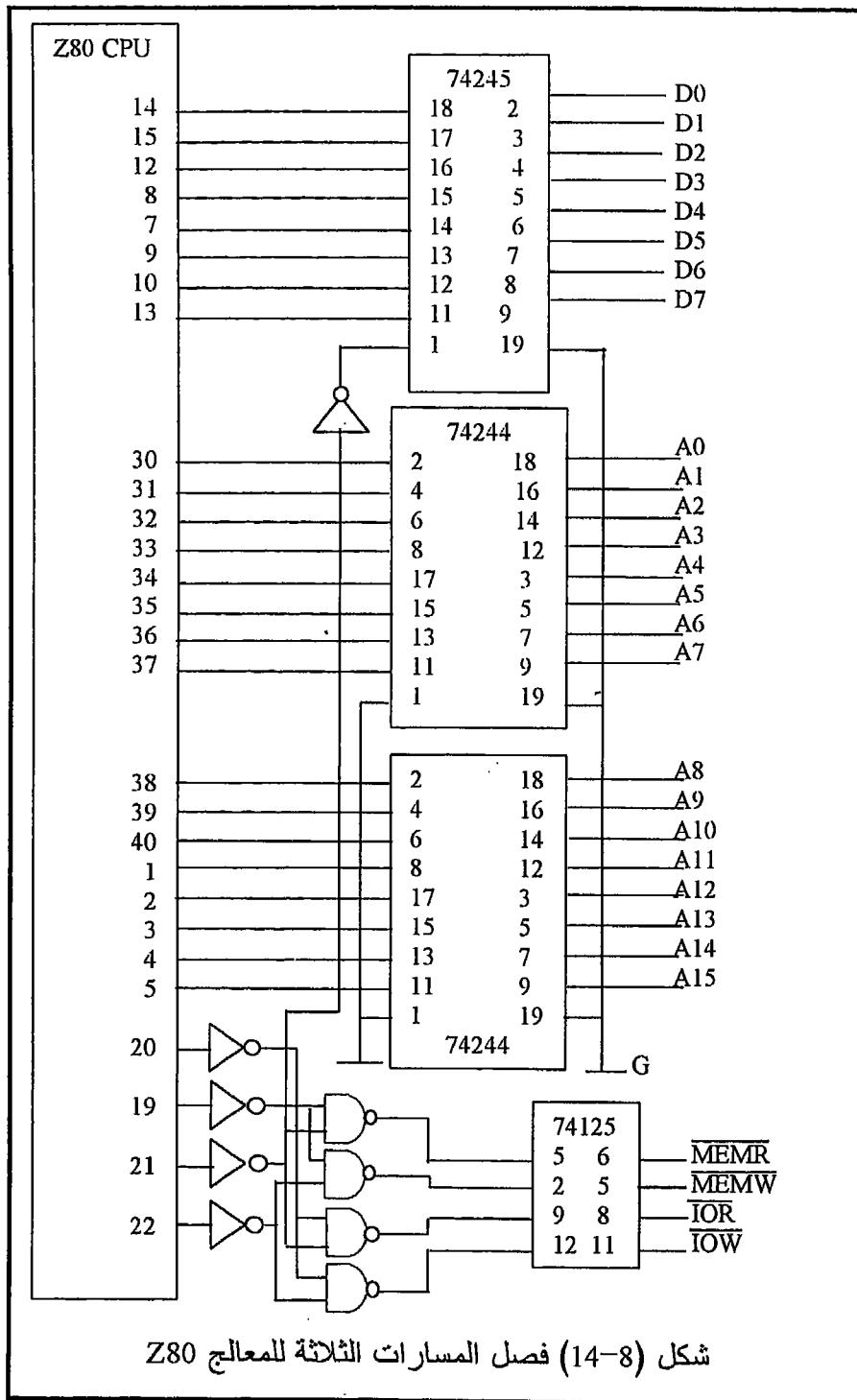
بالقاء نظرة سريعة على أطراف المعالج Z80 كما في شكل (8.13) سنكتشف أن عملية المزج الزمني time multiplexing بين مسارات العنوانين والبيانات التي كانت موجودة في المعالج 8085 غير موجودة هنا ، ولكن كل مسار متاح بصورة منفصلة عن الآخر ولذلك فإن عملية التهيئة هنا ستكون أبسط من قبل . لذلك فإن كل ما سنحتاجه هنا هو عملية فصل buffer لهذه المسارات بغرض الحماية وتوفير التيار اللازم للأجهزة المحيطة . شكل (8-14) يبين المعالج Z80 وقد تم فصل مساراته الثلاثة ، العنوانين والبيانات والتحكم . بالنسبة لمسار العنوانين فقد استخدمت الشريحة 74244 التي تتكون من ثمانى بوابات ثلاثة المنطق لهذا الغرض ، ولقد تم استخدام شريحتان منها لتحقيق عملية الفصل لـ 16 خطًا في مسار العنوانين . لاحظ أن خطوط التحكم في الخرج بالنسبة لـ 16 خطًا قد تم توصيلهما بالأرضى مباشرة مع العلم أن هذه الخطوط يمكن توصيلها على الطرف HOLD القادم من المعالج لتحقيق عملية انفصال المعالج عن المسارات الثلاثة التي سنشرحها في موضع قادم .



بالنسبة لمسار البيانات للمعالج Z80 فقد استخدمنا نفس الشريحة 74245 التي سبق استخدامها لعملية فصل مسار البيانات في المعالج 8085 . هذه الشريحة كما عرفناها من قبل تتكون من ثمانى بوابات ثلاثة المنطق ثنائية الاتجاه أى تسمح بعملية فصل buffer الإشارات التي تمر في اتجاهين . لقد تم التحكم فى اتجاه البيانات عن طريق توصيل الطرف \overline{DIR} رقم 1 فى الشريحة 74245 بالطرف \overline{RD} رقم 21 فى المعالج Z80 بحيث عندما يكون الطرف \overline{RD} فعال (0) فإن الشريحة 74245 ستسمح بمرور البيانات من الأجهزة المحيطة إلى المعالج . وأما عندما يكون الطرف \overline{RD} غير فعال (1) فإن البيانات ستتم فى الاتجاه الآخر .



شكل (8-14) يبين أيضاً كيفية الحصول على خطوط التحكم \overline{MEMR} و \overline{MEMW} و \overline{IOR} و \overline{IOW} باستخدام دوائر NAND . لاحظ أن الطرف MREQ يكون فعالاً (0) عندما يتعامل المعالج مع الذاكرة سواء بغرض القراءة منها أو الكتابة فيها . وأما الخط \overline{IORQ} فيكون فعالاً (0) عندما يتعامل المعالج مع بوابات الإدخال أو الإخراج سواء بغرض القراءة أو الكتابة أيضاً .



شكل (14-8) فصل المسارات الثلاثة للمعالج Z80

لاحظ أن خطوط التحكم الأربع تكون فعالة عندما تكون صفر . بذلك تكون قد انتهينا من تهيئة جميع مسارات المعالج Z80 وأصبحت جاهزة لعملية المواجهة مع الأجهزة المحيطة .

6-8 تمارين

1. ما هو المقصود من تهيئة المسارات ؟ ولماذا تحتاج المسارات إلى تهيئة ؟
 2. الأجهزة المحيطة بالمعالج إما أن تكون أجهزة إدخال (ترسل ، تستقبل) المعلومات (من ، إلى) المعالج ، أو أجهزة إخراج (ترسل ، تستقبل) المعلومات (من ، إلى) المعالج . أختر كلمة مناسبة مما بين القوسين .
 3. مسار البيانات ثنائي الاتجاه ، أى أن الإشارة عليه تكون خارجة من المعالج على بعض الخطوط ، وداخلة إليه على البعض الآخر (صح ، خطأ) أختر ؟
 4. مسار العناوين أحادى الاتجاه يحمل إشارة من الأجهزة المحيطة إلى المعالج (صح ، خطأ) أختر ؟
 5. الذاكرة التي يستطيع معالج من المعالجات تبلغ 64 كيلوبايت لأن (مسار البيانات ، مسار العناوين) 16 بت ، أختر إجابة ؟
 6. لماذا تحتاج لخطوط التحكم MEMR و IOW ؟
 7. لماذا تحتاج لخطوط التحكم IOR و MEMW ؟
 8. عند تنفيذ الأمر STA E100 في المعالج 8085 أي الخطوط التالية سيكون فعالا : MEMR و IOR و IOW ؟ أختر الخط الصحيح .
 9. عند تنفيذ الأمر MOV M,A في المعالج 8085 أي الخطوط التالية سيكون فعالا: MEMR و IOR و IOW ؟ أختر إحدى الإجابات .
 10. عند تنفيذ الأمر IN 00 في المعالجين 8085 و Z80 أي الخطوط التالية سيكون فعالا: MEMR و MEMW و IOR و IOW ؟ أختر إحدى الإجابات .
 11. عند تنفيذ الأمر OUT 00 في المعالجين 8085 و Z80 أي الخطوط التالية سيكون فعالا: MEMR و MEMW و IOR و IOW ؟ أختر إحدى الإجابات .
- .12

xxx:	MVI A,89H STA E100 IN 00 OUT 00 LXI H, E100 MOV B,M JMP xxx 8085 Program	xxx:	LD A,89H LD (E100), A IN 00 OUT 00 LD HL, E100 MOV B,M JP xxx Z80 Program
------	---	------	--

البرنامج السابق عبارة عن حلقة لا نهائية ، ارسم شكل الإشارة مع الزمن على كل خط من الخطوط $\overline{\text{MEMR}}$ و $\overline{\text{MEMW}}$ و $\overline{\text{IOR}}$ و $\overline{\text{IOW}}$ في أثناء تنفيذ هذا البرنامج ؟

13. لو نفذنا برنامج السؤال 12 مرة واحدة ، أختر الإجابة الصحيحة في كل مما يلى :

. عدد نبضات الفعالية للخط $\overline{\text{MEMR}}$ سيكون (17 ، 13 ، 19)

. عدد نبضات الفعالية للخط $\overline{\text{MEMW}}$ سيكون (17 ، 1 ، 9)

. عدد نبضات الفعالية للخط $\overline{\text{IOW}}$ سيكون (17 ، 17 ، 9)

. عدد نبضات الفعالية للخط $\overline{\text{IOR}}$ سيكون (17 ، 1 ، 9)

14. إذا كان الخط $\overline{\text{WR}}=0$ والخط $\overline{\text{IO/M}}=1$ فإن ذلك يعني (قراءة ، كتابة) (فى ، من) (ذاكرة ، بوابة إدخال ، بوابة إخراج) أختر الإجابة الصحيحة ؟

15. إذا كان الخط $\overline{\text{RD}}=0$ والخط $\overline{\text{IO/M}}=1$ فإن ذلك يعني (قراءة ، كتابة) (فى ، من) (ذاكرة ، بوابة إدخال ، بوابة إخراج) أختر الإجابة الصحيحة ؟

16. ما المقصود بالمشاركة الزمنية بين خطوط البيانات وخطوط العنوانين فى المعالج 8085 ؟ وما الهدف منها ؟

17. هل هذه المشاركة موجودة في المعالج Z80 ؟

18. اشرح دور الخط ALE في عملية فصل إشارة العنوانين عن إشارة البيانات على الخطوط AD0-AD7 في المعالج 8085 ؟

19. اشرح كيفية الحصول على خطوط التحكم الأربع في المعالج Z80 ؟

الفصل التاسع

مواجهة الذاكرة

Memory Interfacing

٩-١ مقدمة

إن ذاكرة الكمبيوتر تكون عادة ذاكرة إلكترونية أى أن طريقة مسح المعلومة وحفظها يتم إلكترونياً ، وذلك على العكس من الأنواع الأخرى من الذاكرة مثل شرائط الكاسيت والأقراص المغنة والتى يتم مسح المعلومة عليها مغناطيسياً ، ونحن في هذا الفصل لن نتعرض للتركيب الإلكتروني للذاكرة ولكن كل ما يهمنا هنا هو معرفة كيفية توصيل أو مواجهة المعالج مع شرائح الذاكرة .

إن دليل التليفونات يمكن النظر إليه على أنه نوع من أنواع الذاكرة التي تسجل فيها المعلومات في صورة حروف هجائية ، وهذا النوع يماثل في الخواص ذاكرة القراءة فقط ROM والتي تستخدم في الحاسوبات ، وحيث أنه يمكن قراءة المعلومات من الدليل ولكن لا يمكن تغييرها فكذلك يمكن قراءة المعلومات من ال ROM ولا يمكن الكتابة فيها ولذلك سميت بذاكرة القراءة فقط Read Only Memory . ذلك على العكس من شريط الكاسيت أو القرص الممعنط حيث يمكن تخزين المعلومات عليهما كما يمكن مسحها أو تغييرها في أي لحظة مثلها في ذلك مثل ذاكرة القراءة والكتابة في الحاسوبات والتي سميت عرفاً بذاكرة الاتصال العشوائي Random Access Memory; RAM . أى نظام من نظم الحاسوبات لابد وأن يحتوى على كل من النوعين من أنواع الذاكرة (RAM و ROM) حيث تحتوى ال ROM على الثوابت والبرامج المهمة لتشغيل نظام الحاسوب والتي يمنع المستخدم من الدخول عليها نظراً لخطورة التغيير أو التعديل فيها ، وأما ال RAM فهي الذاكرة التي تكون متاحة للمستخدم ليقوم فيها بتخزين جميع بياناته أو برامجه في حالة تعامله مع النظام . لاحظ أن ال ROM من أهم خواصها أنه عند انقطاع القدرة (الكهرباء) أى عند إطفاء النظام فإن جميع ما بها من معلومات لا تفقد ولكن تظل محفوظة ، وذلك على العكس من ال RAM التي تفقد كل محتوياتها بمجرد إطفاء النظام ، ولذلك فإنه قبل إطفاء النظام لابد من تخزين محتويات ال RAM التي تحتاجها على ذاكرة مستديمة مثل الشرائط أو الأقراص المغنة . شكل (٩-١) يبين مناظرة بين ذاكرة الحاسوبات والذاكرة بمعناها العام .

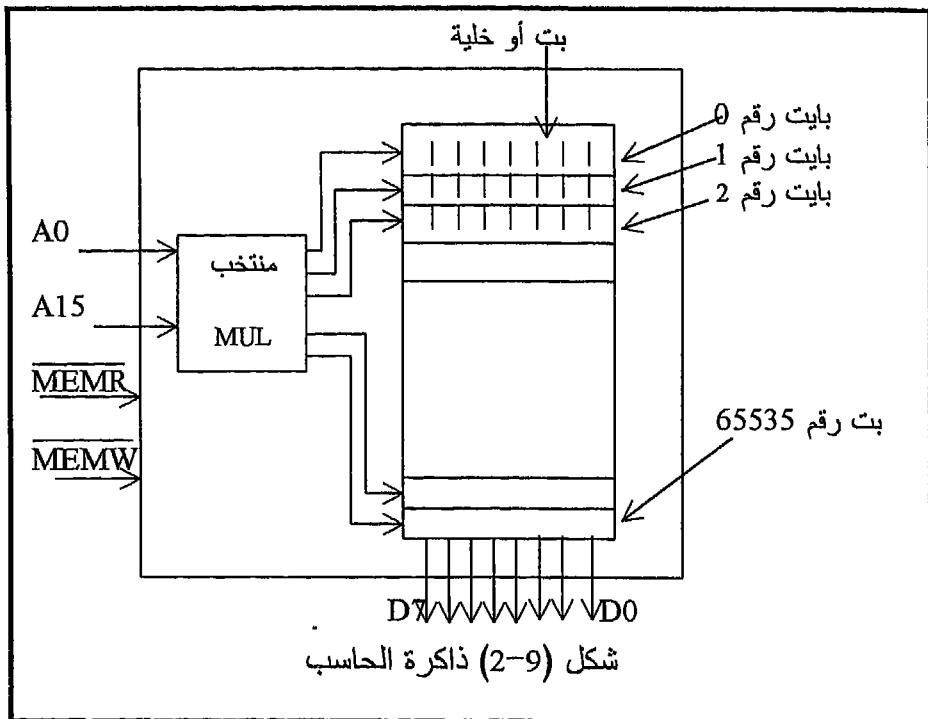
إن كل ذاكرة مهما كانت لابد وأن هناك طريقة معينة تسهل عملية استدعاء المعلومات منها ، فمثلاً من دليل التليفونات يمكنك الوصول إلى رقم تليفون أي شخص عن طريق التسلسل الأبجدي للأحرف ، أما على شريط الكاسيت فيمكنك تحديد مكان المعلومة عن طريق عدد الشريط ، وأما في ذاكرة الحاسوب فإن كل مكان من أماكن الذاكرة محدد بعنوان وهذا العنوان عبارة عن شفرة من الواحد والأصفار توضع على مسار العنوانين فتسبب عملية تشتيط أو إثارة لمكان واحد فقط من أماكن الذاكرة ليصبح جاهزاً للتعامل معه بواسطة المعالج .

الذاكرة عامة	ذاكرة الكمبيوتر
Pick up كلها أمثلة على الذاكرة التي يمنع الكتابة فيها ولكن يسمح فقط بقراءتها.	ROM تحتوى البرامج والثوابت التي يمنع المستخدم من الوصول إليها أو العبث بها نظراً لأهميتها لتشغيل النظام ويمكنه فقط أن يقرأها.
شريط الكاسيت وشريط الفيديو يمكن التسجيل فيها وقراءة محتوياتها ، كما يمكن المسح أو الإضافة لأى جزء فيها .	RAM يمكن للمس تخدم أن يقرأ محتوياتها ويسجل فيها ما يشاء ، ويمسح ويضيف في أي مكان فيها .
يتم بناؤها من أوراق مثل دليل التليفون ، أو شرائط من مواد مغناطيسية مثل شرائط الكاسيت والفيديو .	يتم بناء هذه الذاكرة عادة من أشباه الموصلات ، وهي في العادة عبارة عن قلابات flip flops .

شكل (9-1) مناظرة بين ذاكرة الحاسب والذاكرة عامة

2-9 أساسيات بناء ذاكرة الحاسب

شكل (9-2) يبين أساسيات بناء الذاكرة . يتكون مسار العنوانين القادر من أي واحد من المعالجات التي ندرسها الآن من 16 خطأ أو 16 بت ، ولذلك فإن كمية الذاكرة التي يستطيع المعالج أن يتعامل معها تقدر بـ 2^{16} أي 65536 بait مرتبة كما لو كانت أرفف في دولاب وكل رف من هذه الأرفف يمثل بait وكل بait أو رف تتكون من ثمانى بتات أو خلايا كما في شكل (9-2) . كل واحدة من هذه البيانات تعرف بعنوان خاص بها ولذلك فإن أول بait عنوانها هو (صفر) وأخر بait أو آخر رف في هذا الدولاب عنوانه هو 65535 وذلك في النظام العشري . عند التعامل مع أي بait من هذه البيانات أي القراءة منها أو الكتابة فيها فإن ذلك يكون على البait الكاملة وليس هناك وسيلة للتعامل مع جزء من البait ، أي عدد معين من بتاتها دون الباقي .



لاحظ أن أي شفرة على خطوط العنوانين A0 إلى A15 ستحدد عنوان مكان أو بايت من بآيات الذاكرة في النظام الثنائي ، لاحظ أيضا أنه طالما أن كل بايت تكون من ثماني بتات فإنها تتوافق مع مسار البيانات الذي هو ثماني بتات أو ثمانية خطوط أيضا والذي سيحمل المعلومات من أو إلى هذه البآيات . بعد أن يضع المعالج عنوان البايت التي سيتعامل معها على مسار العنوانين فإنه لابد وأن يحدد طريقة التعامل مع هذه البايت إذا كانت قراءة أو كتابة فيها. إذا كان المعالج يريد الكتابة في الذاكرة فإنه يجعل خط التحكم MEMW (Memory write) وهو خط الإعلان عن الكتابة في الذاكرة فعالا ، أما إذا كان يريد القراءة منها فإنه يجعل خط التحكم MEMR (Memory read) وهو خط الإعلان عن القراءة من الذاكرة فعالا ، لذلك فإنه لزم إضافة هذين الخطين ، MEMW و MEMR في شكل (9-2). تذكر دائماً أن معنى وضع خط فوق اسم أي إشارة يعني أن هذه الإشارة تكون فعالة حينما تكون صفراء Low وهي الحالة الموجدة في الإشارتين MEMW و MEMR. شكل (9-3) يبين بعض العنوانين وشفاراتها الثنائية السعشرية . حاول دراسة هذا الجدول وأضف من عندك شفات لبعض العنوانين الغير مذكورة في الجدول . لاحظ أنه في أثناء البرمجة وفي كل تعاملاتنا مع

العناوين فيما بعد سيكون في الصورة السبعية التي تتكون من أربع خانات كما هو مبين في شكل (3-9).

نظام عشري	نظام سبعيني	الشفرة الموجة على خط العنوان	A15 إلى A0	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0	0000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1	0001	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
2	0010	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
3	0011	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
4	0100	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
5	0101	0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1
.....
.....
5DFB	24059	1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1
.....
ADAC	44460	1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 0
.....
FFFFE	65534	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
FFFF	65535	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

شكل (3-9) عنونة الذاكرة المواجهة للمعالج

شكل (3-2) يبين أيضاً كيفية الاتصال بمكان معين في الذاكرة عن طريق منتخب البيانات الموجود داخل شريحة الذاكرة . يقوم هذا المنتخب Address decoder باختيار أو انتخاب واحد من الخطوط الموجدة على خرجه حيث يتم هذا الاختيار على أساس الشفرة الموجدة على دخله من مسار العنوان وجعله فعالاً وهذا الخط وبالتالي يختار البايت المقابل له وإنما يخرج محتوياتها على مسار البيانات إذا كان الخط MEMW فعالاً أو يدخل محتويات مسار البيانات إلى هذه البايت إذا كان الخط MEMR فعالاً .

إن الزمن المأمور لوضع محتويات أي بايت من بايتس الذاكرة على مسار البيانات أو العكس يسمى زمن الاتصال بالذاكرة Memory access time وهذا الزمن يعتبر خاصية من خواص شريحة الذاكرة حيث يختلف في طوله وقسره على حسب التكنولوجيا والمادة المستخدمة في تصنيع الشريحة ، وعادة يكون هذا الزمن في حدود المائة نانو ثانية (nano second) حيث النانو ثانية تساوى 10^{-9} من الثانية .

إن وحدات قياس سعة الذاكرة في عالم الحاسوب هو الكيلو بايت ك.ب KB . وتم التعارف على أن الواحد كيلوبايت يساوى 1024 بايت بدلًا من 1000 التي

تستخدم دائماً مع تعريف الكيلو في الحياة العملية وذلك لسهولة التعامل مع الرقم 1024 في النظام الثنائي والستة عشرى . كما نعلم فإن شرائط المعالجات التي نتعامل معها لها 16 خطأ للعناوين وبهذا العدد من خطوط العنوانين فإنه يمكن لهذه الشريحة التعامل مع 65536 مكان من أماكن الذاكرة كما رأينا منذ قليل . لاحظ أن 65536 عند قسمتها على 1024 تعطي 64 كيلوبايت لذلك يقال دائماً إن هذه المعالجات يمكنها التعامل مع 64 كيلوبايت ذاكرة . شكل (9-4) يبين علاقة عدد خطوط العنوانين بكمية الذاكرة التي يمكن لأى معالج أن يتعامل معها وكيف أن كمية هذه الذاكرة تتضاعف مع كل زيادة في عدد خطوط العنوانين بمقدار خط واحد .

عدد خطوط العنونة المطلوبة	كمية الذاكرة (بايت)
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	(1 كيلوبايت) 1024
11	(2 كيلوبايت) 2048
12	(4 كيلوبايت) 4096
13	(8 كيلوبايت) 8192
14	(16 كيلوبايت) 16384
15	(32 كيلوبايت) 32768
16	(64 كيلوبايت) 65536

شكل (9-4) مضاعفة كمية الذاكرة بزيادة خطوط العنوانين بمقدار خط واحد

9-3 كيف سنوصل الذاكرة على المعالج؟

يوجد في الأسواق العديد من شرائح الذاكرة التي تختلف من شريحة لأخرى من حيث كمية الذاكرة الموجودة في كل شريحة . فهناك شرائح تحتوى الواحدة منها على واحد كيلوبايت وأخرى تحتوى الواحدة منها على 512 بait وأخرى تحتوى الواحدة منها على 4 كيلوبايت وهكذا ، بل إن هناك شرائح تحتوى الواحدة منها على 64 كيلو بait وأكثر ، السؤال الآن أى هذه الشرائح نستخدم للحصول على الـ 64 كيلو بait التي سنوصلها على المعالج ؟ وما هي أفضل الطرق لتوصيل هذه الشرائح ؟

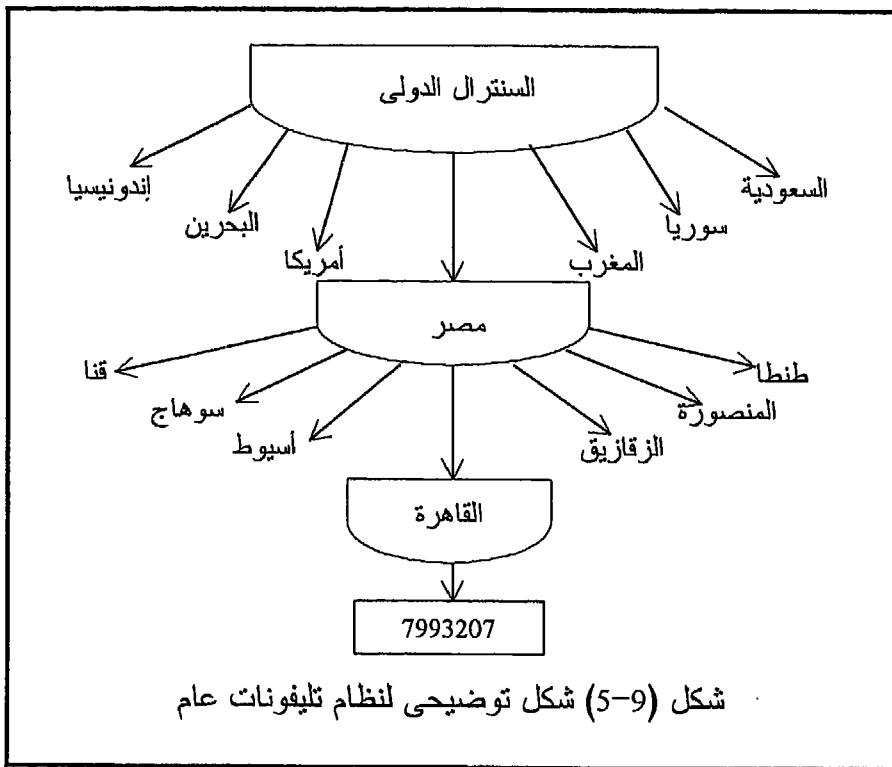
9-3-1 مثال توضيحي

افترض أننا في المملكة العربية السعودية ونريد الاتصال هاتفيًا بالرقم 7993207 الموجود في مصر بمدينة القاهرة . إننا لكي نفعل ذلك لابد أن نضرب أول رقم مصر في السنترال الدولي وهو 002 ثم نضرب رقم مدينة القاهرة في مصر وهو 02 ثم نضرب الرقم الذي معنا ، أى أن الرقم كله الآن أصبح كالتالي 002027993207 . على ضوء ذلك فإننا يمكننا النظر إلى نظام التليفونات في العالم على النحو المبين في شكل (9-5) حيث يتكون هذا النظام من سنترال عالمي يحتوى رقماً لكل دولة من دول العالم وب مجرد ضرب رقم أى واحدة من هذه الدول فإنه يوصلك بسنترال عموم هذه الدولة الذي يحتوى رقم لكل مدينة داخل هذه الدولة ، وب مجرد ضرب رقم أى مدينة من هذه المدن فإنه يوصلك بسنترال هذه المدينة الذي يحتوى جميع الأرقام داخل هذه المدينة ومنها الرقم الذي تريده . السؤال الآن لماذا هذا التعجب في شرح نظام التليفونات وما دخله بموضوع توصيل شرائح الذاكرة على المعالج ؟ إن الشبه كبير جداً بين الاثنين فكما أنك تستطيع النظر لأى رقم تليفون وتقوم بتقسيمه إلى عدة أجزاء حيث جزء منه يمثل الرقم الدولي وجزء يمثل رقم المدينة داخل الدولة وجزء يمثل رقم التليفون داخل المدينة فكذلك يمكن عمل نفس الشيء مع أى عنوان من عنوانين الذاكرة كما سنرى بعد قليل .

9-3-2 نظام بلوكتس الذاكرة

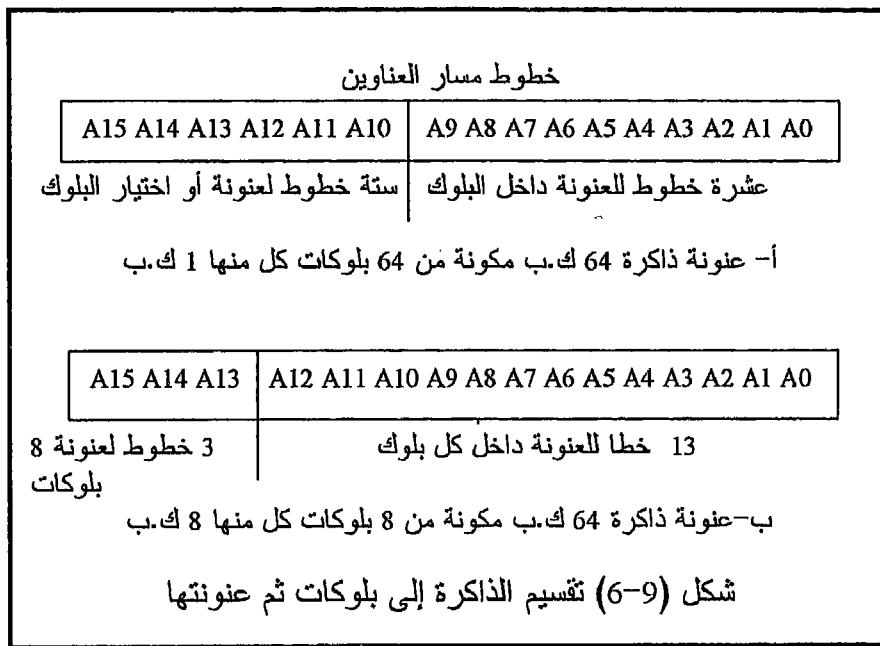
أول ما سنفعله لبناء ذاكرة مقدارها 64 كيلوبايت هو تقسيم هذه الكمية من الذاكرة إلى عدد من البلوكتس يحتوى كل بلوك منها على عدد من الكيلوبايتات . فمثلاً يمكننا تقسيمها إلى 64 بلوك يحتوى كل منها على واحد كيلوبايت ، أو إلى 8 بلوكتس يحتوى الواحد منها على 8 كيلوبايت ، أو إلى 128 بلوك يحتوى الواحد منها على نصف كيلوبايت وهكذا فإن عدد البلوكتس سيترك تماماً للمستخدم

الحرية في تحديده . بمجرد تحديد عدد блوكات سيتعدد فورا كم خطأ من خطوط مسار العنوانين سيستخدم لتمييز блوكات بعضها من بعض وكم خطأ سيستخدم لتمييز البايتات داخل كل بلوك بحيث ستدخل خطوط عنونة أو تمييز البلوكات على منتخب يكون خرجه 2 أس عدد هذه الخطوط بحيث سيذهب كل واحد منها لتنشيط بلوك معين .



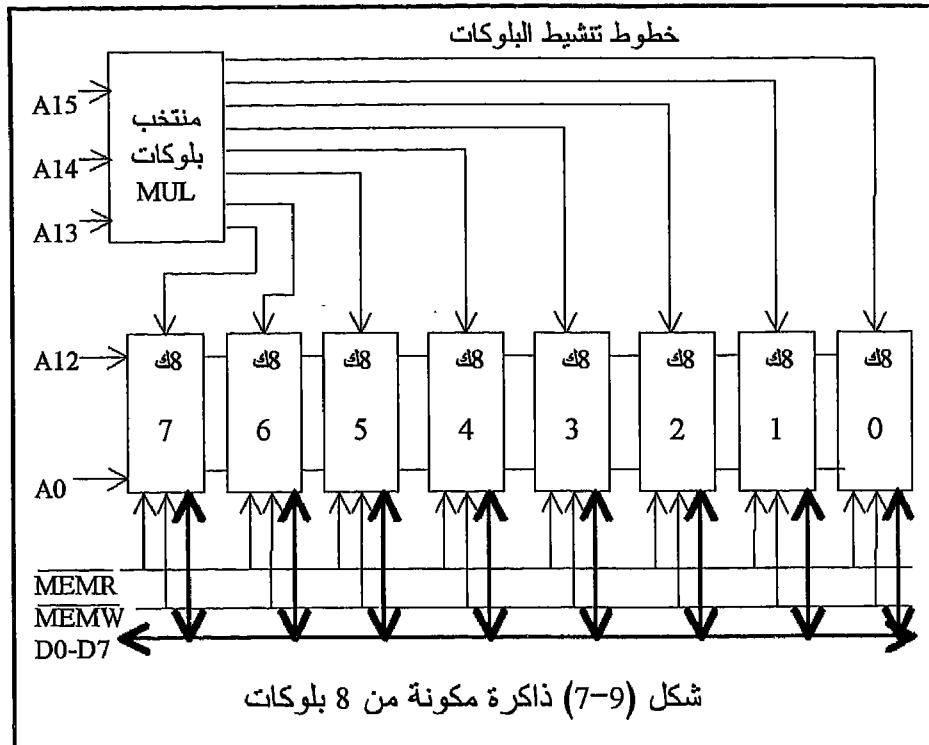
شكل (9-6) يبين تقسيم خطوط العنوانين إلى خطوط عنونة بلوكات وخطوط عنونة بايتات داخل البلوكات في هاتين ، الأولى في حالة تقسيم الذاكرة إلى 64 بلوك يحتوى الواحد منها على واحد كيلوبايت ، والثانية في حالة تقسيم الذاكرة إلى 8 بلوكات يحتوى الواحد منها على 8 كيلوبايت . فى المثال الأول طالما أن وحدة البناء وهى البلوك تساوى واحد كيلوبايت لذلك فإنه يلزم $10^3 = 1024$ خطوط ($= 1024$ واحد كيلوبايت) من ال 16 خطوط الموجودة فى مسار العنوانين لعنونة ال 1024 بايت الموجودة داخل البلوك . أما ال 6 خطوط الباقي من مسار العنوانين ($= 64$) فإنها تستخدم فى عنونة أو اختيار واحد من ال 64 بلوك . المثال الثانى فى شكل (9-6) يبين عنونة ذاكرة من 64 كيلوبايت باستخدام بلوكات كل منها يحتوى على 8 كيلوبايت . لاحظ أن عدد البلوكات

يساوى 64 مقصوما على عدد الكيلوبايتات فى البلوك الواحد . إن خطوط العنونة داخل البلوك ستكون 13 خطأ فى هذه الحالة حيث سيبقى ثلاثة خطوط من ال 16 خطأ لعنونة الثمانية بلوكت . خطوط عنونة البلوكات ستتصل بمنتخب يقوم باختيار واحد من البلوكات الثمانية على حسب الشفرة الموجودة على هذه الخطوط وتسمى الخطوط الخارجية من هذا المنتخب بخطوط اختيار البلوك أو خطوط تنشيط البلوك . شكل (9-7) يبين رسميا تخطيطيا لبناء ذاكرة من 64 كيلوبايت باستخدام بلوكت 8 كيلوبايت .



من شكل (9-7) نلاحظ أن وظيفة منتخب البلوكات هي اختيار واحد من البلوكات على حسب الشفرة الموجودة على خطوط العنوانين A13 و A14 و A15 وتنشطيه أى جعله جاهزا لاستقبال أو إرسال معلومات . لاحظ أيضا أن جميع البلوكات متصلة بخطوط العنوانين A0 إلى A12 لتحديد أى بآيت داخل البلوك الذى تم اختياره سيتم التعامل معها . لاحظ أيضا اتصال كل بلوك بخطوط التحكم MEMR و MEMW و مسار البيانات D0 إلى D7 . مقارنة سريعة بين نظام تقسيم التليفونات العالمى فى شكل (9-5) ونظام تقسيم الذاكرة إلى بلوكت فى شكل (9-7) نرى أن منتخب البلوكات فى شكل (9-7) يلعب نفس الدور الذى يلعبه السنترال الدولى والذى يقوم بتوصيلك إلى دولة معينة على حسب الرقم الداخل له ، فكذاك منتخب البلوكات على حسب الرقم أو الشفرة الموجودة على

الخطوط A13 إلى A15 يقوم بتوصيل المعالج على واحد من هذه البلوكات . بعد قليل سنرى كيفية تقسيم البلوكات إلى شرائح كل منها يحتوى عددا معينا من البايتات والتى تناظر عملية تقسيم الدول إلى مدن كما فى نظام التليفونات .



بمجرد تحديد عدد البلوكات من قبل المستخدم فإن المدى العنوانى لكل بلوك يتحدد على حسب هذا العدد . المدى العنوانى لأى بلوك يتحدد بتحديد عنوان البداية وعنوان النهاية لكل بلوك ، لاحظ أن عنوان البداية لأى بلوك يبدأ من حيث انتهى البلوك السابق له . شكل (9-8) يبين المدى العنوانى للثمانية بلوكات الموجودة في شكل (9-7) وهو المثال الذى سنفترضه دائمًا في عملية بناء الذاكرة .

9-3-3 بناء البلوكات من شرائح

نحن نعلم أن الذاكرة تكون في هيئة شرائح وكل شريحة لها سعة معينة ، فهناك مثلا شرائح سعتها 1 كيلو بايت وأخرى سعتها 2 كيلوبايت أو 512 بايت وأحيانا تجد 64 كيلوبايت على شريحة واحدة ، فما علاقة هذه الشرائح بالبلوكات أو وحدات البناء التي تكلمنا عنها سلفا؟ لما كان البلوك وحدة البناء في هيكل الذاكرة

الكلى فإن الشريحة ستكون وحدة البناء داخل البلوك ، أى أن البلوك يتكون من عدد من الشرائح وهذا العدد سيتوقف على سعة الشرائح المستخدمة ومن المفضل أن تكون جميع الشرائح داخل البلوك الواحد لها نفس السعة . فمثلاً البلوك ذي الثمانية كيلوبايت الذى استخدمناه سلفا يمكن بناؤه من 4 شرائح كل منها 2 كيلو أو من 8 شرائح كل منها 1 كيلو أو من شريحة واحدة سعتها 8 كيلوبايت وهكذا .

رقم البلوك	خطوط العنوان										العنوان ستعشري					
	A15 إلى A0		العنوان													
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	5FFF
3	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	6000
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF
4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	9FFF
5	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	A000
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFF
6	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	C000
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	DFFF
7	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	E000
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

شكل (9-8) المدى العنوانى لكل بلوك من الثمانية بلوكات فى شكل (9-9)

السؤال الآن هو: كيف يتم توصيل هذه الشرائح داخل البلوك الواحد؟ إن طريقة توصيل الشرائح داخل البلوك هي نفسها طريقة توصيل البلوكات للحصول على هيكل الذاكرة . أى أن خطوط العنوان الداخلة للبلوك جزء منها سيستعمل لعنونة البيانات المختلفة داخل الشريحة والجزء الآخر سيستعمل لاختيار أو انتخاب الشريحة . إن عدد الخطوط في كل جزء من هذه الأجزاء سيتحدد على حسب سعة الشرائح المستخدمة ولنفرض البلوك الـ 8 كيلوبايت كمثال . إذا استخدمنا شريحة سعة كل منها 2 كيلوبايت في بناء هذا البلوك فإن كل شريحة ستحتاج إلى 11 خطأ (A0 إلى A10) من الـ 13 خطأ الداخلة إلى البلوك ويتبقى خطان (A11

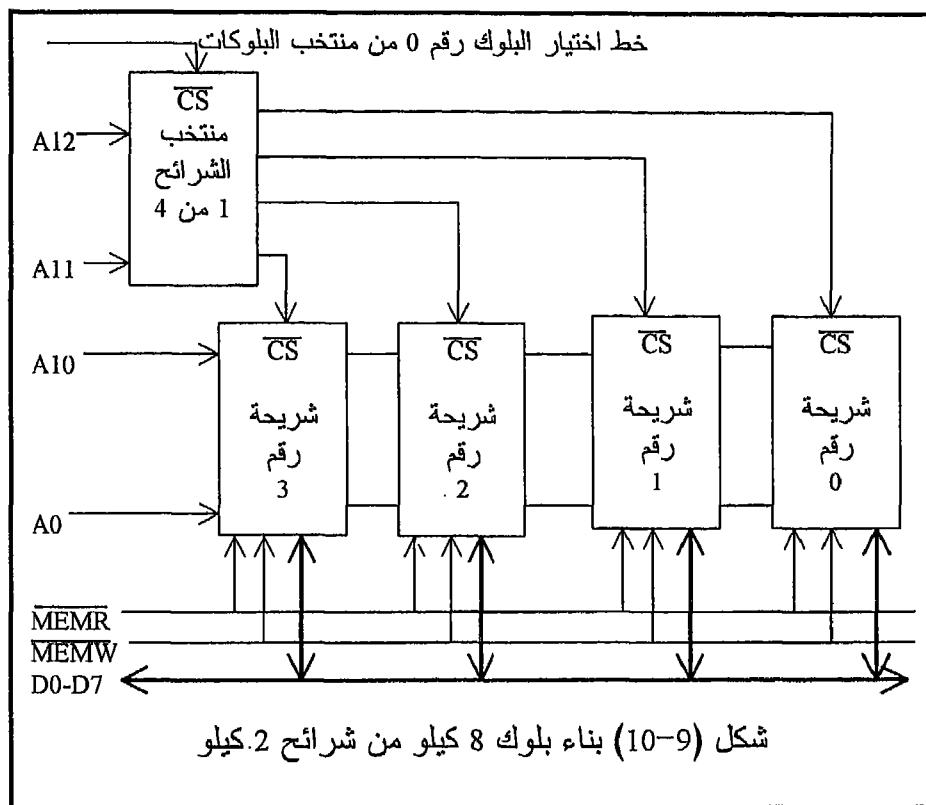
- و A12) يستخدمان في عملية اختيار الشرائح عن طريق منتخب آخر سن سمي به منتخب اختيار الشرائح . لاحظ أنه باستخدام خطين يمكن اختيار واحدة من أربع شرائح حيث 2^2 يساوى 4 . شكل (9-9) عبارة عن مثال لبيان المدى العنوانى لكل شريحة من شرائح البلوك الأول ، وحاول أنت كتابة المدى العنوانى لشريحة واحد من البلوكات الأخرى . من شكل (9-9) نلاحظ الآتى :
- أول عنوان فى أول شريحة فى البلوك هو أول عنوان فى البلوك ، فمثلا فى شكل (9-9) العنوان الأول فى أول شريحة فى أول بلوك هو 0000 .
 - آخر عنوان فى آخر شريحة فى البلوك هو آخر عنوان فى البلوك ، فمثلا فى شكل (9-9) آخر عنوان فى آخر شريحة فى البلوك الأول هو FFFF وهو آخر عنوان فى أول بلوك كما فى شكل (9-8) .
 - خطوط اختيار البلوك A13 و A14 و A15 كانت أصفارا دائمًا لأننا فى البلوك الأول الذى مده العنوانى 0000 إلى FFFF .
 - خطوط اختيار الشريحة ثابتة لا تتغير طول مدى الشريحة ، فالشريحة الأولى خطوط اختيارها كانت 00 والشريحة الثانية خطوط اختيارها كانت 01 وهكذا .
 - أول عنوان فى الشريحة نحصل عليه بأن نضع جميع خطوط اختيار البايت داخل الشريحة (A0 إلى A10) تساوى أصفارا ، وأخر عنوان فى الشريحة نحصل عليه بأن نضع هذه الخطوط تساوى وحيد .

العنوانى	المدى	رقم الشريحة
0000	0	
07FF		
0800	1	
0FFF		
1000	2	
17FF		
1800	3	
FFFF		

شكل (9-9) المدى العنوانى لكل شريحة داخل البلوك الأول

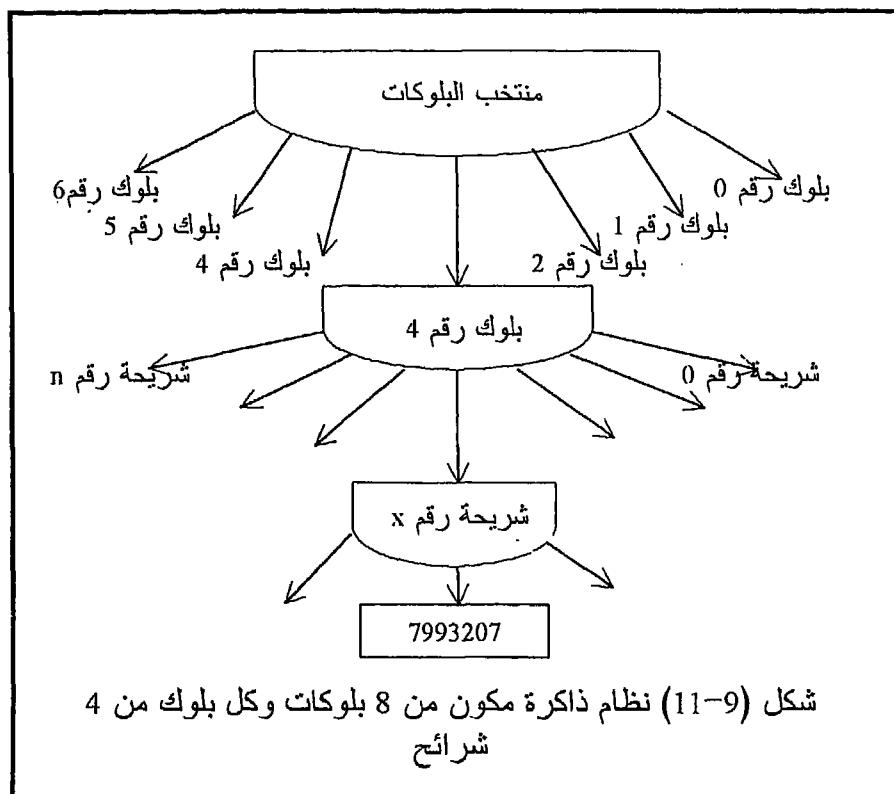
شكل (9-10) يبين التركيب الداخلى لأحد البلوكات الثمانية المبينة فى شكل (9-7) حيث يتكون هذا البلوك من أربع شرائح كل منها 2 كيلوبايت ولذلك فإن كل شريحة من هذه الشرائح لها 11 خطأ للعناوين متصلة بالخطوط A0 إلى A10 القادمة من مسار عناوين المعالج . كذلك فإن كل شريحة كما نرى لابد وأن تكون متصلة بمسار العناوين D0 إلى D7 . إن منتخب الشريحة الموجود فى

شكل (9-10) وبناء على الشفرة الموجودة على دخليه A11 و A12 سيكون واحد فقط من خروجه فعالاً وباقى الخروج خاملة مما سيجعل شريحة الذاكرة المتصلة بهذا الخرج هى فقط الفعالة وأما باقى الشرائح فستكون خاملة ، وعلى ذلك فإن هذه الشريحة وبناء على الشفرة الموجودة على الخطوط A0 إلى A10 ستكون إحدى بالياتها جاهزة للتعامل مع المعالج إما للكتابة أو القراءة على حسب حالة الخطين ، $\overline{\text{MEMR}}$ و $\overline{\text{MEMW}}$. لاحظ أن منتخب الشرائح فى شكل (9-10) لن يكون فعالاً إلا إذا جاءت له إشارة أو نبضة من منتخب блوكات فى شكل (7-9) تخبره أن هذا البلوك قد اختير للتعامل مع المعالج وبذلك يصبح منتخب الشرائح فعالاً .



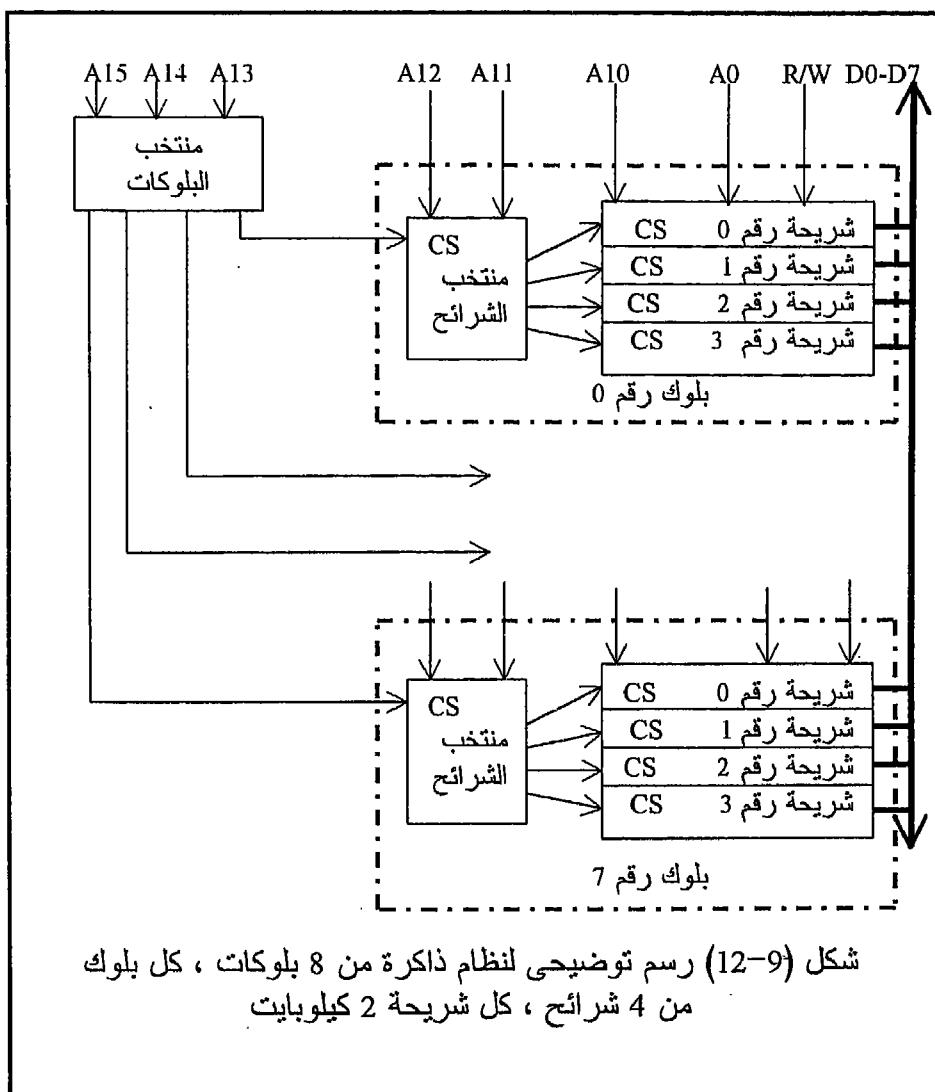
شكل (11-9) يبين نظام ذاكرة مكون من 8 بلوكات وكل بلوك مكون من 4 شرائح وكل شريحة مكونة من 2 كيلوبايت فهل نستطيع الآن مقارنة هذا الشكل بشكل (9-5) الذى يبين رسمياً توضيحاً لنظام تليفونات عالمي؟ إن منتخب البلوكات يناظر السنترال الدولى ومنتخب الشرائح يناظر سنترال الدولة التى تم اختيارها وعنوان البأيت داخل الشريحة يناظر رقم أى تليفون داخل الدولة التى

تم اختيارها بإهمال سنترالات المدن . شكل (9-12) يبين رسمياً توضيحاً لنظام الثمانية بلوكتات الذي نفترضه في شرحنا وذلك حتى يتمكن القارئ من إلقاء نظرة شاملة على نظام الذاكرة بأكمله . لاحظ أن البلوكتات ما بين الأول والأخير في هذا الشكل ما هي إلا تكرار مثل البلوك رقم صفر والبلوك رقم 7 .

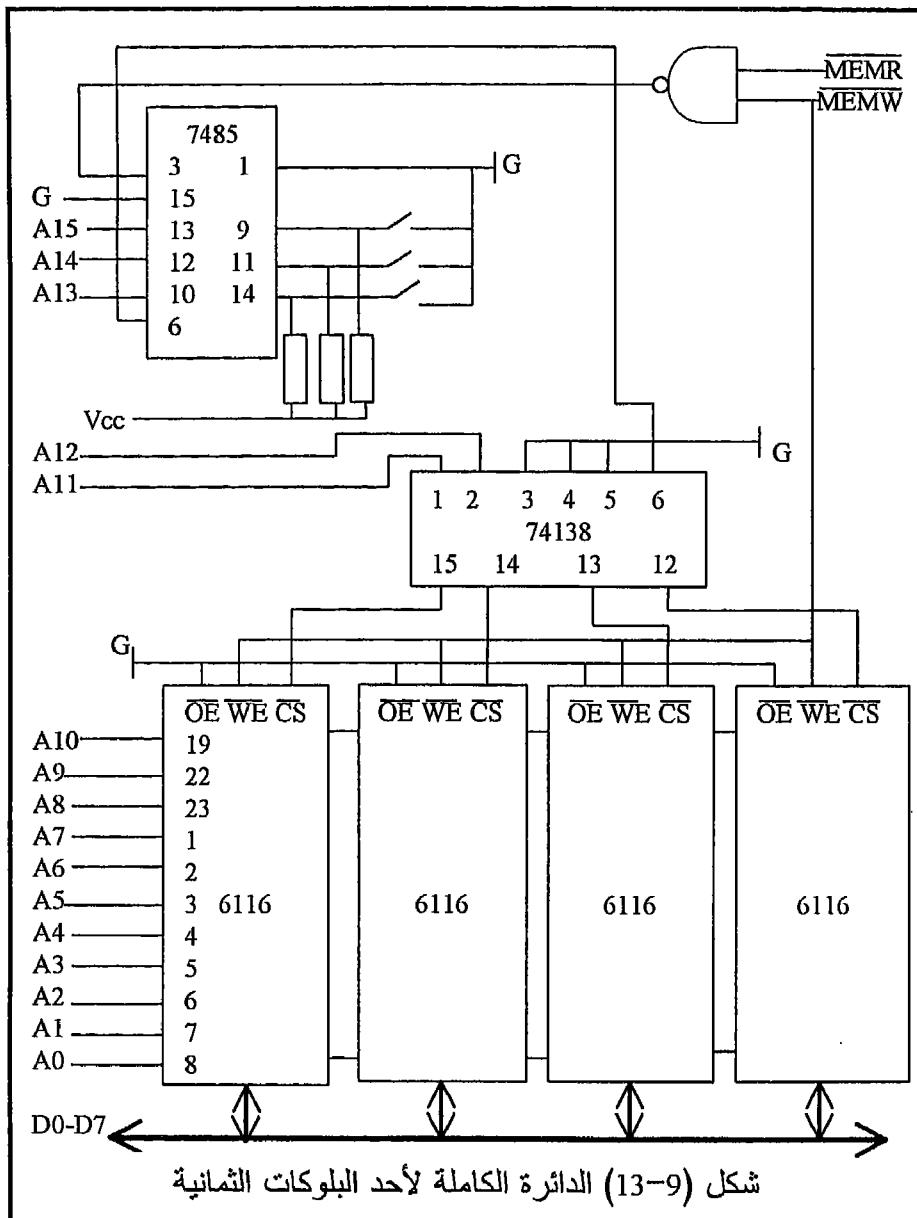


إن نظام بناء الذاكرة باستخدام блокات الذي تم شرحه حتى الآن ليس هو النظم الوحيد الذي يجب أن يؤخذ به في عملية بناء أي نظام ذاكرة ، ولكن من مميزات هذا النظام أنه يعتبر عاماً يمكن تعديله ليناسب أي غرض . فمن الممكن مثلاً أن يبني النظام بحيث يكون كل بلوك على كرت منفصل وتكون الذاكرة في هذه الحالة عبارة عن مجموعة من الكروت التي يستعمل منها ما يلزم للحاجة فقط . أيضاً يمكن استخدام نوعي الذاكرة (إما RAM أو ROM) على أي واحد من هذه الكروت ، فقط يجب الحرص عند توصيل خطوط التحكم ، MEMR و MEMW . خلاصة القول أن نظام الكروت هذا يمكن أن يحور أو يعدل ليتناسب مع أي تصميم مطلوب . شكل (9-13) يبين الدائرة الكاملة لأحد البلوكات السابقة وقد

استخدمنا معه شريحة الذاكرة رقم 6116 التي تحتوى على 2 كيلوبايت استاتيكية . هذه الشريحة متوافقة تماما من حيث وظيفة الأطراف مع الشريحة رقم 2716 التي تحتوى على 2 كيلوبايت EPROM بحيث أنه يمكن استخدام أى واحدة من الشريحتين مكان الأخرى .



شكل (9-12) رسم توضيحي لنظام ذاكرة من 8 بلوکات ، كل بلوک من 4 شرائح ، كل شريحة 2 كيلوبايت



شكل (9-13) الدائرة الكاملة لأحد блوكات الثمانية

إن جميع محتويات هذا الفصل ليست خاصة بمعالج بعينة دون الآخر ولكنها تناسب أي واحد من المعالجات التي هي تحت دراسة في هذا الكتاب ، ولقد رأينا في الفصل السابق كيف حصلنا على المسارات الثلاثة لكل معالج في الصورة المناسبة لعملية المواجهة ولذلك فإننا نلاحظ أن الشرح لم يكن موجهاً إلى أي

معالج بعينة ولكن كل ما قيل في هذا الفصل يمكن تطبيقه مع أي واحد من المعالجات تحت الدراسة في هذا الكتاب .

4- تمارين

1. مطلوب توصيل شريحة ذاكرة EPROM سعتها 2 كيلوبايت على معالج ، ما هي أبسط الطرق لتوصيل هذه الشريحة مع العلم أنها ستكون شريحة الذاكرة الوحيدة ؟
2. مطلوب توصيل شريحتي ذاكرة EPROM و RAM سعة كل منها 2 كيلوبايت على المعالج ، ما هي أبسط الطرق لذلك مع العلم أنهما شرائح الذاكرة الوحيدة الموصولة على المعالج ؟
3. لديك العديد من شرائح الذاكرة RAM و EPROM التي سعة كل منها 1 كيلوبايت ، ارسم نظام ذاكرة متكامل مكون من بلوکات واقتصر أنت عدد البلوکات الذي ستستعمله ؟ اكتب عنوان البداية وعنوان النهاية لكل بلوک وكل شريحة ؟
4. ارسم نظام ذاكرة متكامل مكون من 16 بلوک مستخدما شرائح كل منها 4 كيلوبايت ؟ اكتب عنوان البداية وعنوان النهاية لكل بلوک وكل شريحة ؟
5. أعد السؤال الرابع إذا كانت الشرائح المستخدمة سعة كل منها 4 كيلوبايت أيضا ولكن مسار البيانات لكل منها 4 بتات بدلا من 8 ؟

الفصل العاشر

الإدخال والإخراج

Input and Output

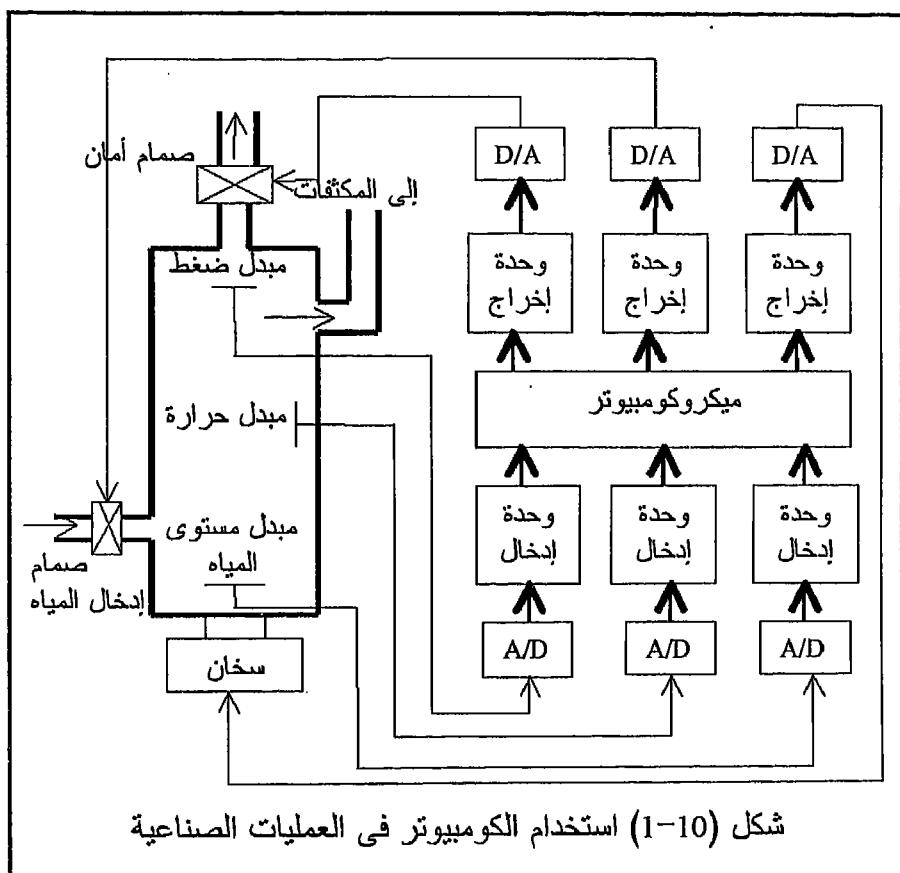
1-10 مقدمة

بالإضافة إلى وحدة التحكم المركزي أى ال CPU والذاكرة بأنواعها فإن الكومبيوتر ودوائر التحكم التي تستخدم المعالجات أو الكومبيوتر لابد وأن تحتوى على وحدات لإدخال وإخراج المعلومات والتى من شأنها تسهيل عملية تبادل المعلومات بين الأجهزة المحيطة والمعالج . الأجهزة المحيطة مثل الطابعات والشاشات والمحولات من رقمى إلى تماثلى والعكس وأى أجهزة خارجية يقوم المعالج بالتحكم فيها أو التعامل معها . لقد رأينا فى الفصل السابق كيفية توصيل المعالج على شرائح الذاكرة سواء كانت RAM أو ROM وستقوم فى هذا الفصل بعرض كيفية بناء وتوصيل المعالج على بوابات الإخراج والإدخال . ونؤكد هنا أيضاً أن محتويات هذا الفصل ليست خاصة بمعالج عينه دون الآخر ولكنها صالحة للاستخدام مع أى واحد من المعالجات التي نستخدمها في هذا الكتاب وبالذات بعد أن رأينا فى الفصل الثامن كيفية الحصول على المسارات المختلفة للمعالجات وتهيئتها . لكي نتم الإحساس لدينا بأهمية الحاجة إلى بوابات الإخراج والإدخال وأين ومتى تكون الحاجة إليها ماسة سنعرض الآتى :

إن وجود شريحة المعالج كوحدة عمليات مركبة داخل الميكروكومبيوتر ليست الوظيفة الوحيدة لمثل هذا النوع من الشرائح . هناك مجال واسع من الاستخدامات والتطبيقات لهذه الشريحة وهو استخدامها كعنصر أساسى من عناصر أنظمة التحكم ، فمثلاً التحكم فى أى عملية كيماوية يمكن لشريحة المعالجات أن تلعب دوراً مهماً فيه ، التحكم فى عمليات خلط البنزين والهواء داخل السيارة مجال من المجالات التي استخدمت هذه الشريحة . إن هذه مجرد أمثلة بسيطة ولكن مجمل القول هو أن شريحة المعالجات تلعب في الوقت الحالى دوراً مهماً جداً في معظم العمليات الصناعية بل وفي الكثير من الأجهزة والآلات التي نستخدمها في حياتنا اليومية .

شكل (10-1) يبين نظاماً بسيطاً لتنقير المياه وقد استخدم الكومبيوتر كعنصر أساسى من عناصر التحكم فيه . إن المياه المطلوب تنقيرها يتم تسخينها باستخدام سخان إلى أن يتحول الماء إلى بخار في أعلى التنك حيث يتم سحب هذا البخار في أنبوبة يمين أعلى التنك حيث يتم تكثيف البخار إلى مياه نقية . إن السخان يجب أن يتوقف عندما تصل درجة حرارة المياه إلى درجة قصوى ويعاد تشغيله إذا وصلت الحرارة إلى حد أدنى ، ويتم ذلك عن طريق ميكروكومبيوتر مخزن فيه كل من الدرجتين العظمى والصغرى ويقوم الكومبيوتر بقراءة درجة حرارة التنك من خرج مبدل (Transducer) لدرجة الحرارة على فترات زمنية محددة ويقارنها بالدرجتين العظمى والصغرى ويقرر إذا كان سيشغل السخان أم يوقفه . بنفس الطريقة يقرأ الكومبيوتر مبدل مستوى المياه في التنك ويقارنها

بأعلى وأقل قيمة لمستوى المياه حيث هاتان القيمتان مخزنتان في الذاكرة ويقرر بناء على ذلك إذا كان سيتم فتح أو غلق صمام إدخال المياه . إن ضغط بخار الماء الموجود في أعلى التنك من الممكن أن يزيد إلى درجة الخطورة ، لذلك فقد تم وضع مبدل للضغط في هذا الجزء من التنك حيث يقوم الميكروكمبيوتر من وقت لآخر بقراءة خرج هذا المبدل ومقارنتها بقيمتين عظمى وصغرى وعلى ضوء هذه المقارنة يقرر إذا كان سيترك صمام الأمان مغلقاً أم يفتحه لفترة محددة من الزمن لتقليل الضغط في أعلى التنك . من ذلك نرى أن مهمة الميكروكمبيوتر هي الاحتفاظ بدرجة حرارة التنك ومستوى المياه فيه وكذلك مستوى الضغط في أعلىه عند القيمة المثلث لكل منها . إن عدد المتغيرات التي يتم التحكم فيها بواسطة الميكروكمبيوتر من الممكن أن يكون أكثر من ذلك بكثير . بل وعادة ما تكون عملية التحكم في متغيرات تnk المياه هذه عبارة عن جزء بسيط من عمليات تحكم أخرى يقوم الكمبيوتر بالتحكم فيها داخل أحد المصانع .

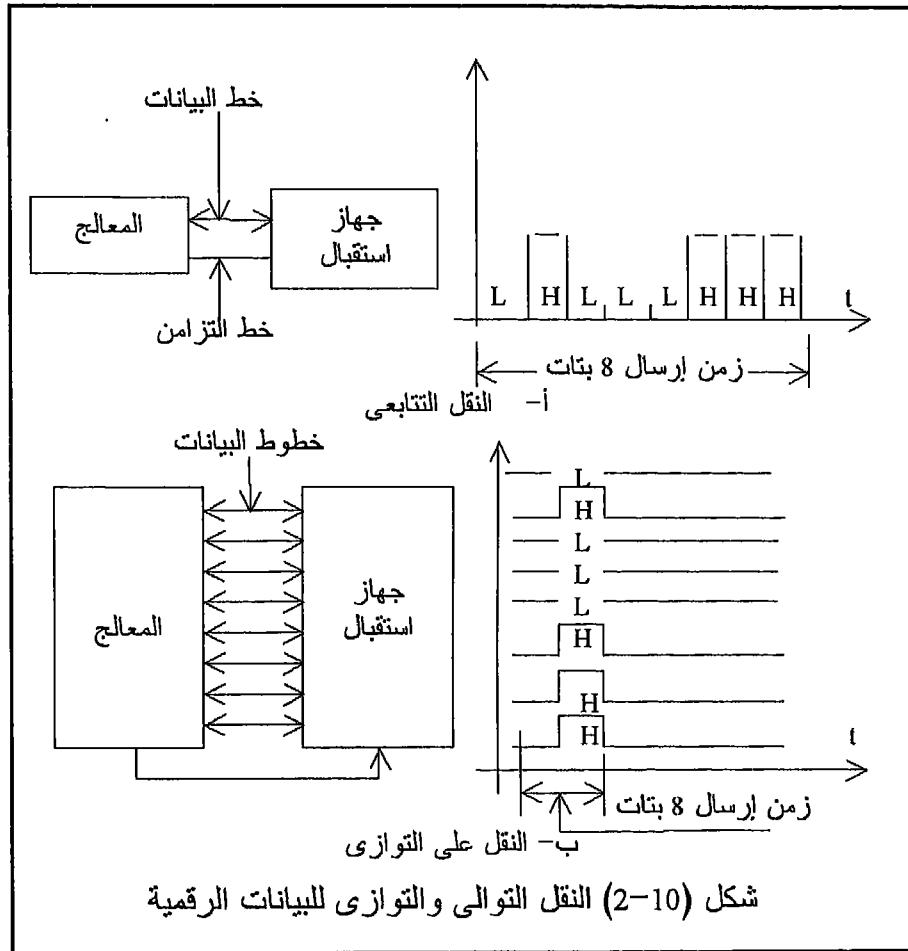


شكل (10-10) استخدام الكمبيوتر في العمليات الصناعية

إن أهم ما يجب ملاحظته في هذا المثال الموجود في شكل (10-1) هو عملية الربط بين عالم الرقمنيات المتمثل في الكمبيوتر حيث تكون جميع الإشارات الموجودة رقمية فقط (Digital) ، وعالم التنازليات أو التماثليات المتمثل في تلك المياه وإشارات التحكم فيه حيث تكون جميع هذه الإشارات تماثلية (Analog) . فمثلاً لكي يعمل السخان لابد من توصيله بجهد عال (220 فولت مثلاً) ، وكذلك لكي تفتح أو تغلق أيها من الصمامات لابد من استخدام هذا الجهد العالى ، إذن كيف يتم هذا الربط ؟ أو بمعنى آخر كيف تحول الإشارات الرقمية الخارجة من الكمبيوتر إلى إشارات تماثلية أو جهد عال يتم به تشغيل السخان مثلاً؟ وكيف يتم أخذ الإشارة الخارجية من أي من المبدلات وهي إشارة تماثلية غالباً ما تكون ضعيفة ، كيف يتم تحويلها إلى إشارة رقمية يمكن للكمبيوتر أن يتعامل معها ؟ جميع هذه الأسئلة وغيرها سيتم الإجابة عنها في هذا الفصل والفصول القادمة . قبل الإجابة عن هذه الأسئلة سنذلل بعض الوقت في التعرف على كيفية إدخال وإخراج المعلومات إلى ومن الكمبيوتر بافتراض أن هذه المعلومات موجودة في الصورة الرقمية .

10-2 طرق إرسال واستقبال المعلومات الرقمية

إن المعلومات التي يتم إدخالها أو إخراجها من أو إلى الكمبيوتر تكون إما في صورة تتابعية Serial data أو في صورة متوازية Parallel data وكل من الصورتين له مميزاته والاستخدامات الخاصة التي تحتاج إليه بالضرورة . في الطريقة التتابعية يتم إرسال المعلومات من وإلى الأجهزة الخارجية على خط واحد فقط ولا يرسل على هذا الخط إلا بت bit واحدة فقط في نفس وحدة الزمن وهي ال Clock ، بحيث أنه لكي يتم إرسال معلومة من ثمانية باتات مثلاً فإننا نحتاج إلى زمن مقداره ثمانى نبضات تزامن لكي نرسل هذه المعلومة . أما في الطريقة المتوازية فإن المعلومات يتم ارسالها من وإلى الكمبيوتر على أكثر من خط واحد ، وعادة ما يكون عدد هذه الخطوط يساوى عدد الخطوط فى مسار البيانات للكمبيوتر الذى يتم التعامل معه وفي هذه الحالة فإننا لكي نرسل معلومة من ثمانى باتات مثلاً سنحتاج إلى ثمانية خطوط متوازية بحيث ترسل كل بات على خط منفصل من هذه الخطوط وبالطبع فإنه في هذه الحالة سترسل جميع هذه الباتات في خلال نبضة تزامن Clock واحدة فقط ، لذلك فإن هذه الطريقة أسرع بكثير في إرسال المعلومات من الطريقة التتابعية السابقة . شكل (10-2) يبين رسمياً توضيحاً لطرق إرسال المعلومات بالطريقتين التتابعية والمتوازية .



شكل (10-2) النقل التوالى والتوازى للبيانات الرقمية

إن عملية إدخال أو إخراج معلومة من أو إلى الكمبيوتر تتكون من جزأين ، الجزء الأول منها هو برنامج Software يقوم الكمبيوتر بتنفيذـه ، والجزء الثاني هو دائرة Hardware يتم بناؤها لكي تقوم بدور الوسيط بين الكمبيوتر والأجهزة الخارجية . لكتابـة برنامـج يقوم بإدخـال أو إخـراج معلومـات من أو إلى الكمبيوترـ فإذاـن هـناك أـيضا طـريقـتين : الطـريقـة الأولى يتمـ فيها استـخدـام الأمر IN لإـدخـال المعلومـة وأـمر OUT لإـخـراجـها وـالـتـى تـسـمى بـطـرـيقـة خـرـيـطة الإـدخـال والإـخـراج للـإـدخـال والإـخـراج OUT Input Output mapped I/O . الطـريقـة الثانية وـفـيـها يتمـ استـخدـام خـرـيـطة ذـاـكـرـة الـكـوـمـبـيـوـتـر ولـذـلـك تـسـمى هـذـه الطـريقـة Memory mapped I/O حيثـ يـعـالـمـ الجـهـازـ الـخـارـجـيـ كـمـا لوـ كانـ باـيـتـ Byteـ منـ باـيـاتـ الـذاـكـرـةـ ولـذـلـكـ يـمـكـنـ معـ هـذـهـ الطـريقـةـ استـخدـامـ أوـامـرـ المعـالـجـ العـادـيـةـ مـثـلـ الأوـامـرـ STAـ، LDAـ، MOVـ،

الأمرین OUT, IN فقط . سیأتى شرح هذه الطريقة فيما بعد وكذاك سنرى أن الدائرة أو ال Hardware الذى يستخدم مع كل من الطريقتين لا يختلف كثيرا .

3-3 الطريقة الأولى من طرق الإدخال والإخراج باستخدام الأمرین OUT, IN

هذه الطريقة من طرق الإدخال والإخراج لا تستعمل مع المعالج MC6800 على الإطلاق حيث لا تحتوى قائمة أوامر هذا المعالج على الأمرین IN و OUT ولكن تستخدم معه طريقة خرائط الذاكرة التى سنشرحها فى الجزء القادم إن شاء الله .

كما نعلم فإن الأمرین OUT, IN يتكون كل منهما من 2 بait ، البایت الأولى هي شفرة الأمر ، أما البایت الثانية فتحتوى على رقم بوابة الإخراج أو الإدخال التى سيتم التعامل معها . كما نعلم أيضا فإن البایت تتكون من ثمانى بتات ولذلك فإنه يمكن تشفير عدد من بوابات الإدخال أو بوابات الإخراج يصل إلى 256 بوابة ⁽²⁾ . عندما يقوم المعالج بتنفيذ الأمر OUT أو الأمر IN فإن رقم البوابة يتم وضعه على الثمانية خطوط الأولى من مسار العناوين ، فمثلا عند تنفيذ الأمر OUT 05 حيث 05 هى رقم أو عنوان بوابة الإخراج فإن المعالج يضع الرقم 05 على مسار العناوين كالتالى :

A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	1	0	1	

أما محتويات مسجل التراكم وهى المعلومة المطلوب إخراجها فيتم وضعها على مسار البيانات تمهدًا لانتقاطها بواسطة بوابة الإخراج فى اللحظة المناسبة . كذلك الحال بالنسبة للأمر IN فمثلا عند تنفيذ الأمر IN 3F فإن رقم بوابة الإدخال وهو 3F يقوم المعالج بوضعه على مسار العناوين كالتالى :

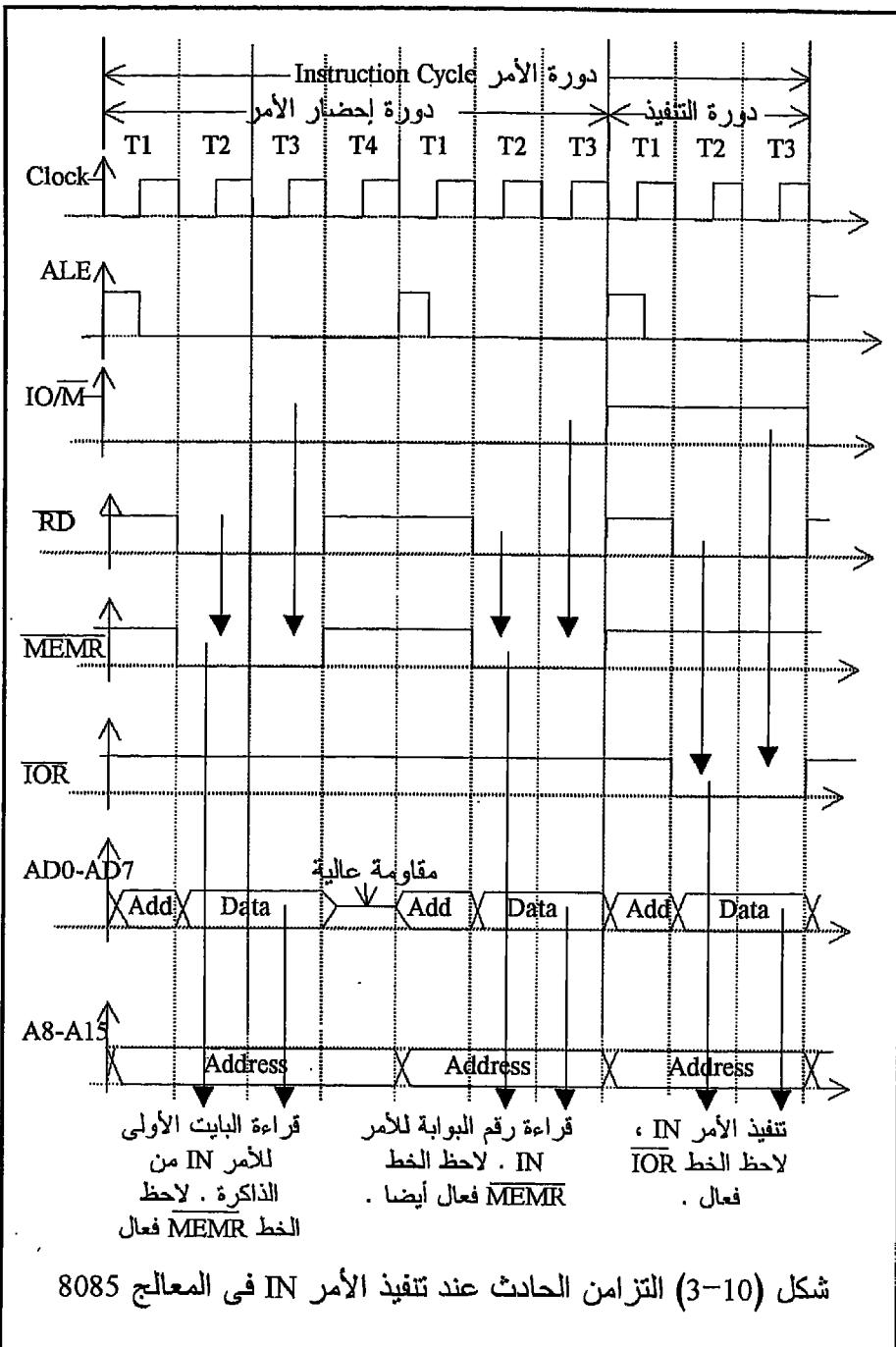
A7	A6	A5	A4	A3	A2	A1	A0
0	0	1	1	1	1	1	

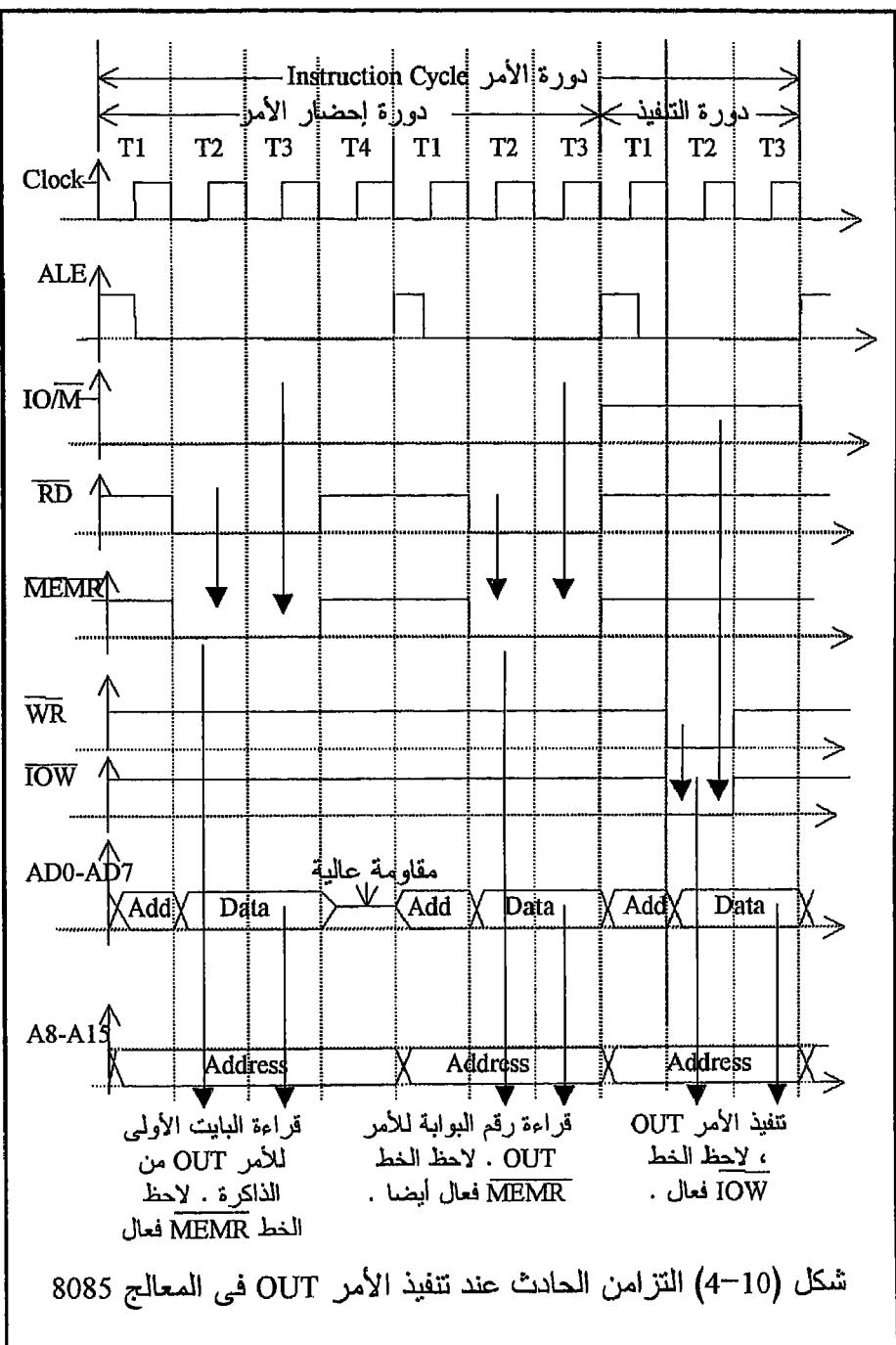
أما المعلومة المراد إدخالها فتنقل من مسار البيانات الذى يكون متصلًا ببوابة الإدخال فى هذه اللحظة إلى مسجل التراكم ، ويجب التأكيد هنا على أنه فى أثناء تنفيذ الأمر OUT فإن الخط IOW من مسار التحكم يكون فعالاً أي صفرًا ، وكذلك فى أثناء تنفيذ الأمر IN فإن الخط IOR من مسار التحكم يكون فعالاً ويساوى أيضًا صفرًا . لذلك فإن الخطين IOW و IOR مهمان جدا في بناء كل من دائرتى الإدخال والإخراج كما سنرى ، ولقد رأينا فى فصل سابق كيفية الحصول على هذه الخطوط مع كل معالج نقوم بدراسته فى هذا الكتاب .

الشكلان (10-3) و (10-4) يبينان التزامن الموجود على جميع خطوط التحكم التي تتأثر بالأمرتين IN و OUT في حالة المعالج 8085 . هذان الشكلان يستحقان التأمل والتدقيق من قبل أي قارئ حيث أنه يمكن منها ملاحظة وفهم الكثير من طريقة تفازد المعالج لأى أمر والتزامن الذي يحدث بين إشارات التحكم المختلفة . شكلا (10-5) و (10-6) يبيّنان التزامن الموجود على جميع خطوط التحكم التي تتأثر بالأمرتين IN و OUT في حالة المعالج Z80 وهذان الشكلان يستحقان التدقيق أيضاً من جميع القراء للاحظة مدى التشابه بين معظم هذه الإشارات .

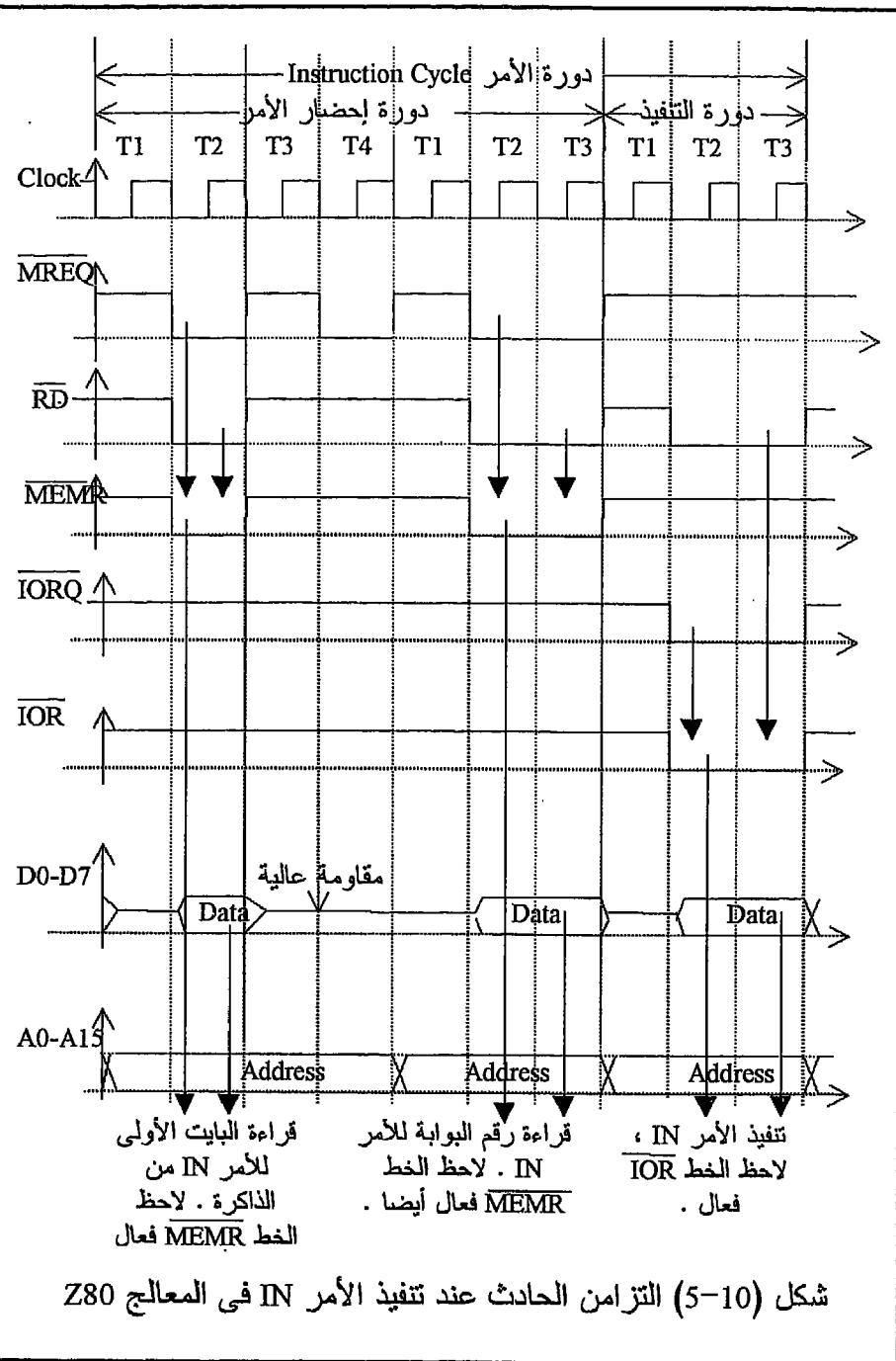
10-3-1 دائرة بوابة الإخراج Output port

شكل (10-7) يبيّن رسمًا تخطيطيًا لبوابة إخراج . كما رأينا فإن كلاً من الشريحة Intell 8085 و Z80 يمكنها التعامل مع 256 بوابة إخراج يمكن ترقيمها من بوابة رقم صفر إلى البوابة رقم 255 . لذلك فقد خصصت المفاتيح S1 إلى S8 لاختيار رقم للبوابة ووضعه في الصورة الثانية باستخدام هذه المفاتيح . عند تنفيذ الأمر OUT فإن رقم البوابة كما ذكرنا من قبل يظهر على الثمانية خطوط الأولى من مسار العنويين A0 إلى A7 . هنا تكون مهمة مقارن العنويين AddressComparator حيث يقارن الرقم الموجود على مسار العنويين مع الرقم الموضوع بالمفاتيح ، فإذا تطابق الرقمان يقوم مقارن العنويين بإعطاء إشارة خرج تدل على ذلك ، وإذا لم يتطابقا يظل خرجه خالما دون تغيير . تستخدم إشارة الخرج القادمة من مقارن العنويين لتشغيل ماسك الخرج Output latch ذي الثمانية بิตات كما هو مبيّن بشكـل (10-7) . لاحظ أن دخل ماسك الخرج متصل بمسار البيانات والذي توضع عليه المعلومة المراد إخراجها في أثناء تنفيذ الأمر OUT ، وفي حالة وجود إشارة الخرج القادمة من مقارن العنويين يقوم ماسك الخرج بنقل المعلومة الموجودة على دخله إلى خرجه حيث يمكن إظهارها على شاشة أو مظاهرات أو الاستفادة منها في أي غرض من الأغراض . الذي يهمنا الآن هو أن المعلومة الموجودة داخل المعالج وبالتحديد في مسجل التراكم تم إخراجها على ماسك حيث من هنا تبدأ الاستفادة بها . لاحظ أيضًا أن الخط IOW تم استخدامه كخط تشغيل مع مقارن العنويين بحيث لا يعمل مقارن العنويين إلا إذا كان هذا الخط فعالاً أي يساوى صفرًا .

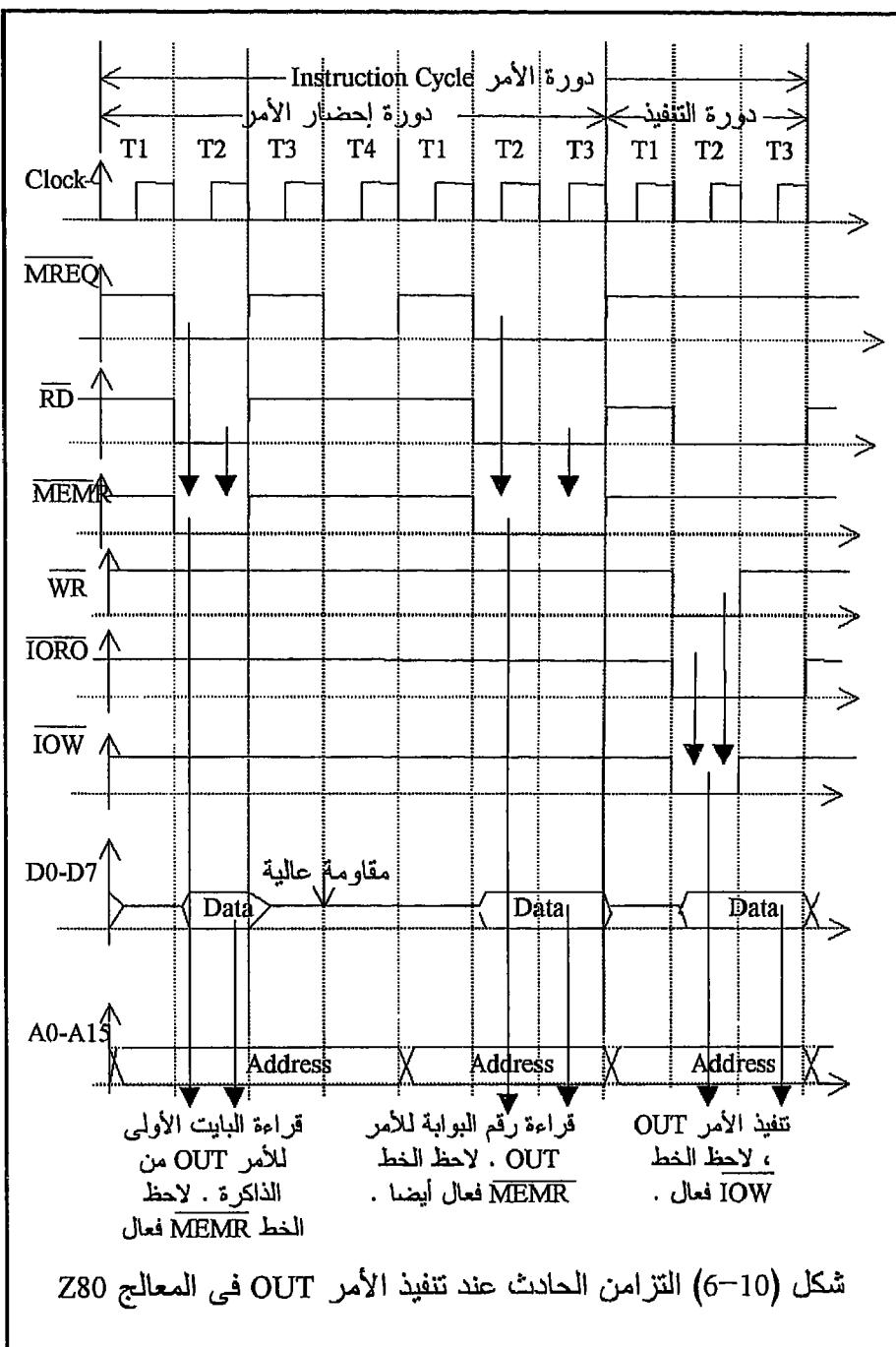




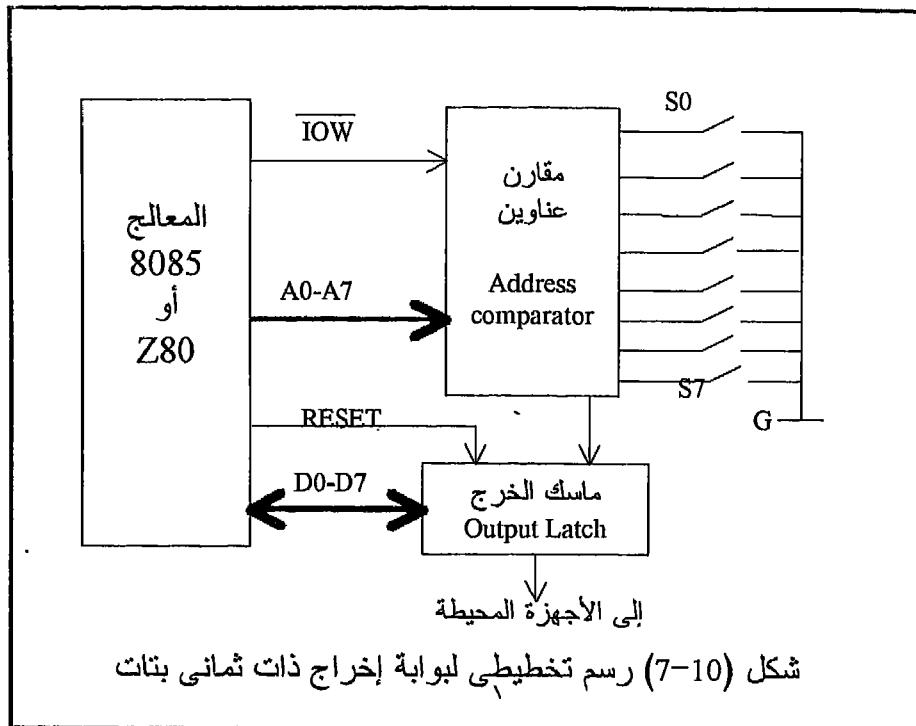
شكل (4-10) التزامن الحادث عند تنفيذ الأمر OUT في المعالج 8085



شكل (5-10) التزامن الحادث عند تنفيذ الأمر IN في المعالج Z80



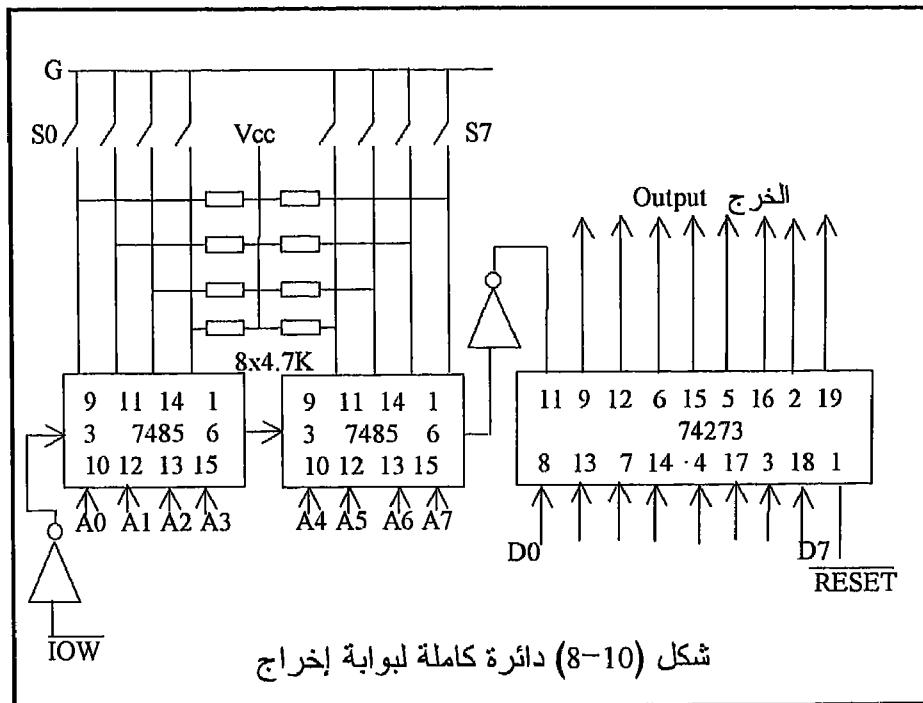
شكل (6-10) التزامن الحادث عند تنفيذ الأمر OUT في المعالج Z80



شكل (10-7) رسم تخطيطي لبوابة إخراج ذات ثمانى بิตات

شكل (10-8) يبين الدائرة الكاملة لبوابة الإخراج . الشريحتان 7485 كل منهما عبارة عن مقارن ذى أربع بิตات حيث يكون خرج الشريحة على الطرف 6 يساوى High إذا كانت كل بت من الدخل A تساوى نظيرتها من الدخل B وكان دخل الشريحة على الطرف 3 يساوى High أيضا . لذلك فإن الإشارة IOW تم إدخالها على الطرف 3 للشريحة الأولى وتم توصيل خرج هذه الشريحة (الطرف 6) إلى دخل الشريحة الثانية (الطرف 3) ، بذلك تعمل الشريحتان كمقارن ذى ثمانى بิตات يؤخذ خرجه من الطرف 6 للشريحة الثانية . هذا الخرج من الشريحة الثانية يستخدم كنبضات تزامن $Clock$ للشريحة 74273 وهى الماسك وذلك بعد عكسه . الشريحة 74273 عبارة عن ماسك ذى ثمانى بิตات وعند عبور التزامن (الطرف 11) من Low إلى High تنتقل الإشارة الموجودة على الدخل إلى الخرج مباشرة . الطرف 1 للشريحة 74273 خاص بالتصفير CLR أو $Clear$ بحيث عندما يكون هذا الطرف Low يصبح كل خرج الشريحة يساوى صفر ولذلك فقد تم توصيل هذا الطرف على الـ $RESET$ لضمان أن يكون خرج الماسك يساوى صفرًا عند بداية التشغيل . لاحظ أن الاستفادة من الخرج تبدأ من هنا حيث يمكن توصيل خرج الماسك على محول رقمى/تماثلى أو على مظهر وذلك على حسب

التطبيق الذى ستستخدم فيه هذه البوابة . لقد كان من الممكن استخدام منتخب decoder بدلًا من مقارن العناوين ولكن ميزة استخدام المقارن أنه يمكن تغيير رقم أو عنوان البوابة على حسب الطلب باستخدام المفاتيح S0 إلى S7 .

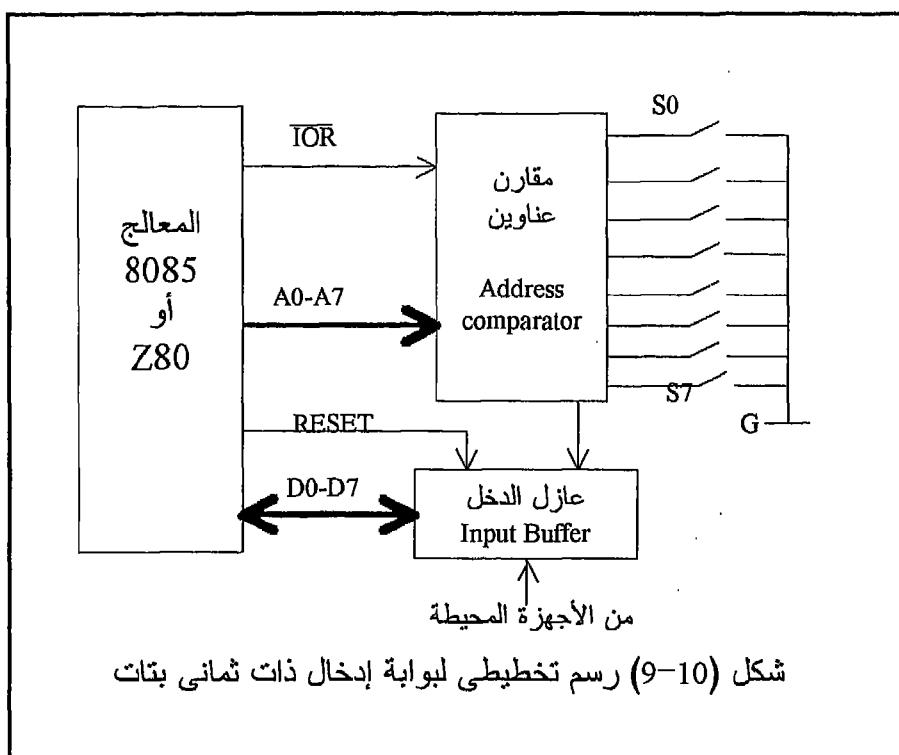


شكل (10-8) دائرة كاملة لبوابة اخراج

10-3-2 دائرة بوابة الإدخال Input port

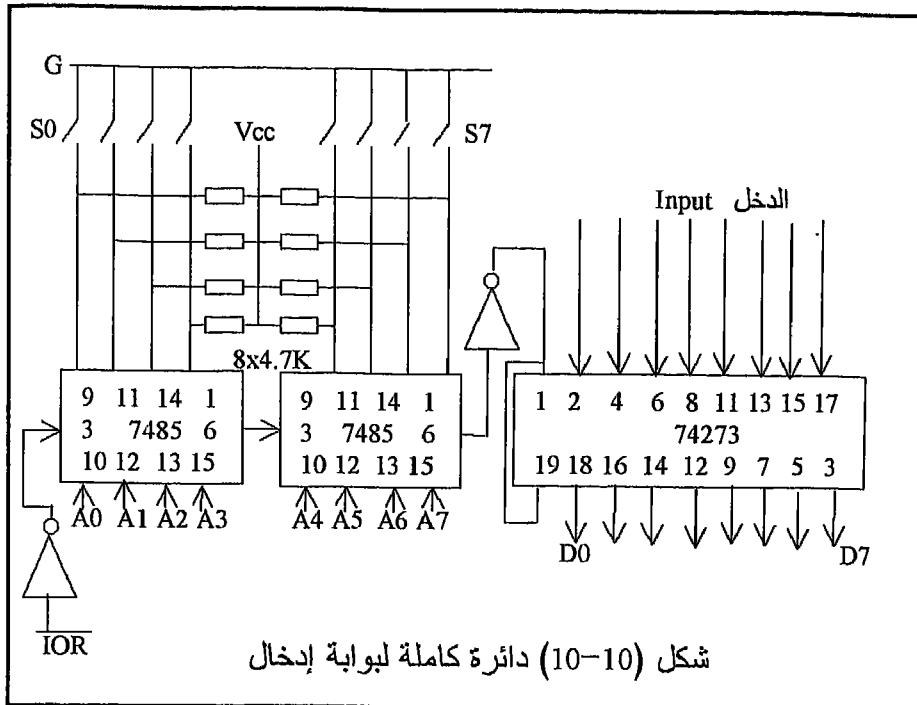
شكل (10-9) يبين رسمًا تخطيطياً لبوابة إدخال . كما علمنا فإن كلا من الشريحة Z80 و Intel 8085 يمكنهما التعامل مع 256 بوابة إدخال والتي يمكن ترقيمها أو عنونتها من البوابة رقم صفر حتى البوابة رقم 255 . لذلك فقد خصصت المفاتيح S1 حتى S8 لاختيار رقم للبوابة ووضعه في الصورة الثنائية باستخدام هذه المفاتيح . عند تنفيذ الأمر IN فإن رقم البوابة كما ذكرنا من قبل يظهر على الثمانية خطوط الأولى من مسار العناوين A0 إلى A7 ، لذلك فإن مهمة مقارن العناوين تكون هي مقارنة الرقم الموجود على مسار العناوين مع الرقم الموضوع بالمفاتيح ، فإذا تطابق الرقمان يعطى إشارة خرج وإذا لم يتطابقا يظل الخرج خاملاً غير فعال . تستخدم إشارة الخرج القادمة من مقارن العناوين لتشغيل عازل Buffer ذي ثمانى بتات الذى رقمه 74244 ، انظر إلى محتويات هذه الشريحة في الفصل الثامن . إن دخل العازل متصل بالمصدر

الذى تأتى منه المعلومة المراد إدخالها إلى المعالج ، لاحظ أنه عندما يكون مقارن العناوين خاماً أي أن خرجه غير فعال فإن العازل أيضاً سيكون غير فعال وسيكون خرجه عبارة عن مقاومة عالية حتى لا يسبب تحميلاً لمسار البيانات Short circuit في حالة اتصال مسار البيانات بأى جهاز آخر وقد عرفنا ذلك في دراستنا لبوابات المنطق الثلاثي . لقد تم استخدام الخط \overline{IOR} كخط تشغيل مع مقارن العناوين بحيث لا يعمل مقارن العناوين إلا إذا كان هذا الخط فعالاً أي يساوى صفراء .



شكل (10-9) رسم تخطيطي لبوابة إدخال ذات ثمانى بتات

شكل (10-10) يبين الدائرة الكاملة لبوابة إدخال . لاحظ مدى التناقض بين هذه الدائرة ودائرة بواية الإخراج . الفرق الأساسي هو استخدام ماسك للخرج في حالة بواية الإخراج لمسك المعلومة التي سيتم إخراجها أما في حالة بواية إدخال فيستخدم عازل Buffer لعزل مسار البيانات عن المصدر الذي تدخل منه المعلومة .



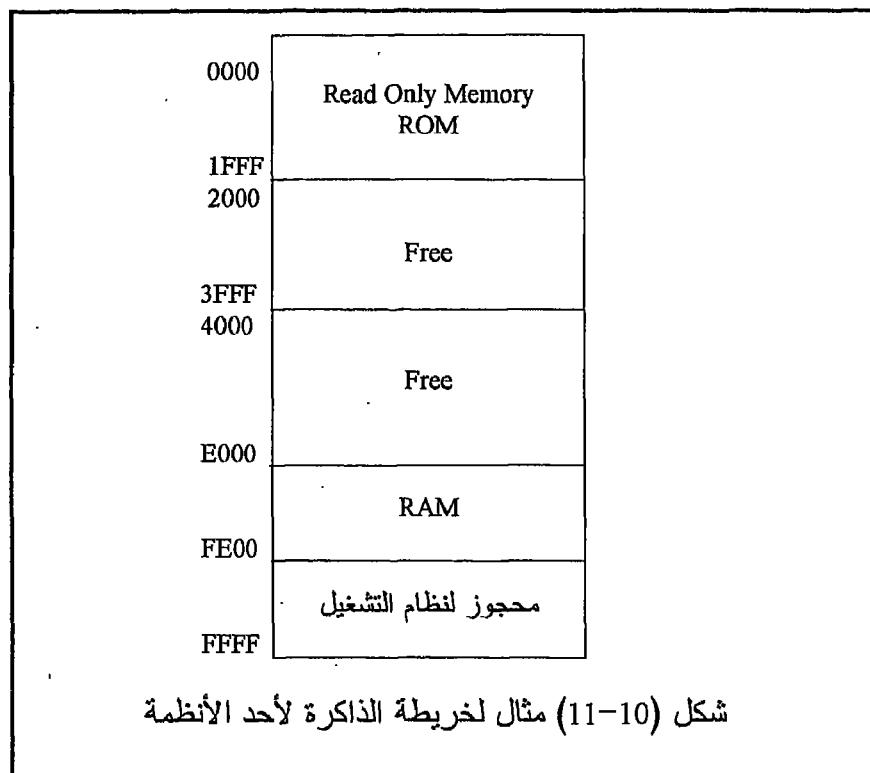
شكل (10-10) دائرة كاملة لبوابة إدخال

4-4 الطريقة الثانية من طرق الإخراج والإدخال باستخدام خريطة الذاكرة Memory mapped I/O

إن خريطة الذاكرة لأى حاسوب هي عبارة عن رسم تخطيطي يبين فيم تستخدمن كل بait من بايتات الذاكرة ابتداء من أول بait إلى آخر بait فى الذاكرة . فمثلا خريطة الذاكرة الخاصة بالميكروكومبيوتر الذى نستخدمه مع أمثلة هذا الكتاب والموضحة فى شكل (10-11) فيها البايتات ابتداء من 0000 إلى 1FFF مشغولة بذاكرة قراءة فقط Read only memory , EPROM ، أما نظام التشغيل الخاص بهذا الميكروكمبيوتر فيشغل الذاكرة ابتداء من البايت FC00 إلى FFFF. انظر إلى شكل (10-11) لتعرف فيما تستخدم باقى الذاكرة . يمكن كما سترى استخدام طريقة خرائط الذاكرة للإدخال والإخراج مع أى معالج ولكن مع المعالج MC6800 فإن هذه الطريقة هي الوحيدة التى يمكن استخدامها معه وذلك لأن هذا المعالج ليس لديه أوامر إدخال وإخراج مثل أبناء جيله 8085 و Z80 .

في هذه الطريقة من طرق الإدخال والإخراج تعامل بوابة الإدخال أو الإخراج كما لو كانت بايت من بايتات الذاكرة ولذلك فإن جميع أوامر المعالج الخاصة

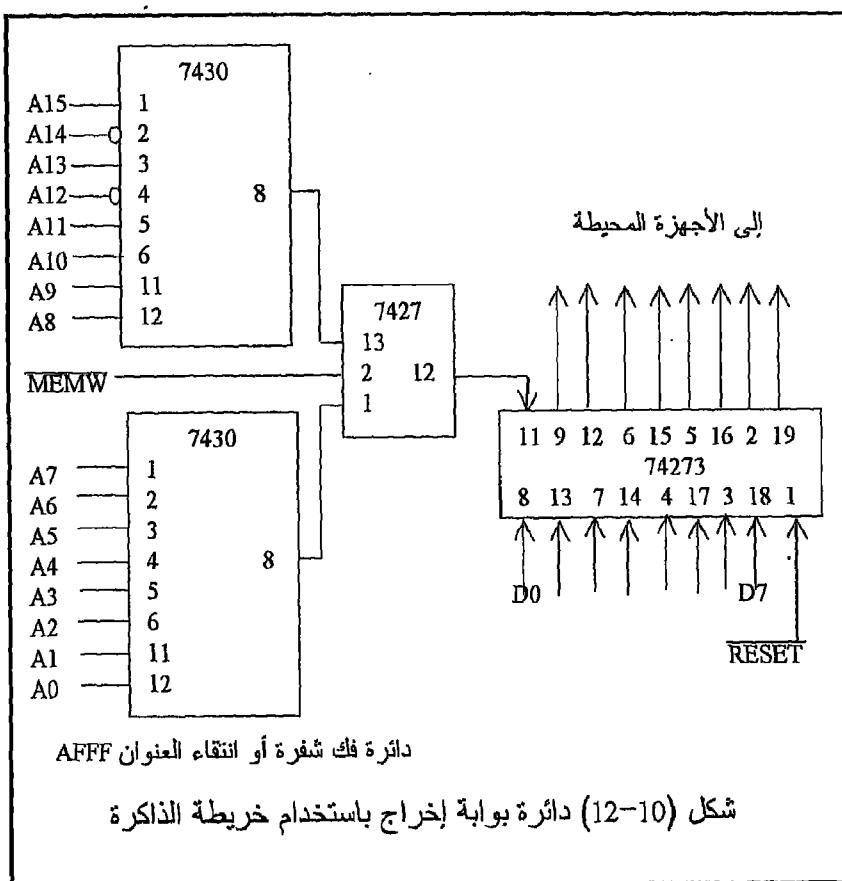
بالتعامل مع الذاكرة يمكن استخدامها في هذه الحالة . لكن نستخدم بایت من بایتات الذاكرة في هذا الغرض فإنه يجب التأكد من أن هذه البايت غير مستخدمة في أي غرض من أغراض التخزين الأخرى ويمكن معرفة ذلك طبعاً بعد النظر في خريطة الذاكرة وانتقاء البايت المناسب لذلك بحرص شديد . بالنظر في خريطة الذاكرة فإننا سنبتعد عن مساحات الذاكرة المستخدمة لنظام التشغيل والمستخدمة كذاكرة قراءة فقط وكذلك المستخدمة كذاكرة تخزين Read/Write memory وسنلجاً لمساحة المحددة بالعناوين A000 إلى BFFF مثلاً وسوف نستخدم أي مكانين في هذه المساحة أحدهما كبوابة إدخال والآخر كبوابة إخراج . إنه في هذا المجال يجب أن نذكر جيداً أنه عندما تقوم شريحة المعالج بتفيذ أي أمر من الأوامر التي تتعامل مع الذاكرة فإن عنوان المكان الذي سيتم التعامل معه يوضع على مسار العناوين (16 خط) وفي هذه الحالة فإن خطى التحكم MEMR و MEMW يوضحان ما إذا كان هذا التعامل سيكون بغرض القراءة من الذاكرة أو الكتابة فيها على حسب ما يكون أي من الخطين فعالاً . لذلك فإن كلاً من هذين الخطين سيلعبان دوراً مهماً في عملية التشفير الخاصة ببوابة الإدخال أو بوابة الإخراج كما سنرى فيما يلى :



10-4-1 دائرة بوابة إخراج باستخدام خريطة الذاكرة

افتراض مثلاً أنه بالنظر في خريطة الذاكرة الخاصة بالمايكروكمبيوتر الذي نستخدمه وجدنا أن مساحة الذاكرة التي تبدأ من المكان A000 إلى المكان BFFF غير مستخدمة لأى غرض من أغراض التخزين ، لذلك فإننا سنستخدم أحد أماكن هذه المساحة كبوابة إخراج ولتكن مثلاً المكان رقم AFFF . شكل (10-12) يبين دائرة مقترحة لهذه البوابة . عند تنفيذ الأمرين :

MVI A,05
STA AFFF



بواسطة المعالج فإن الأمر الأول سيضع الرقم 05 في مسجل التراكم والأمر الثاني سيتسبب في وضع العنوان \overline{AFFF} على مسار العنوانين (A0 إلى A15) وسيجعل الخط \overline{MEMW} فعالا ، ثم يضع محتويات مسجل التراكم على مسار البيانات ، لذلك فإنه بالنظر إلى شكل (10-12) فإن خرج فاكك الشفرة سيكون فعالا مما يسبب نبضة تزامن Clock pulse للماسك (الشريحة 74273) الذي يقوم بإخراج محتويات مسار البيانات على الخرج وذلك هو المطلوب من أي بوابة إخراج حيث يمكن في هذه الحالة الاستفادة من الخرج أو إظهاره على شاشة أو مظهر من أي نوع كما فعلنا مع بوابة الإخراج سابقا . لاحظ أنه يمكن استخدام مقارن للعنوانين كما في الأجزاء السابقة بدلا من فاكك الشفرة أو منتقي العنوان \overline{AFFF} الموجود في شكل (10-12) .

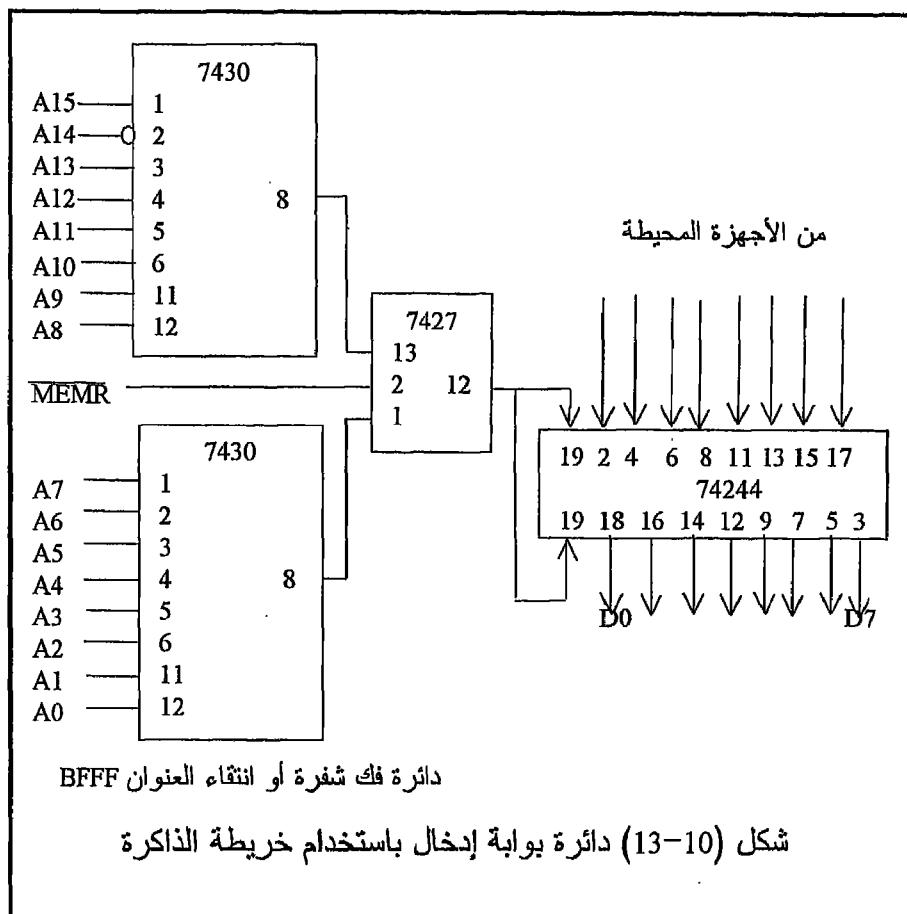
4-4-2 دائرة بوابة إدخال باستخدام خريطة الذاكرة

افتراض أننا سنستخدم المكان \overline{BFFF} من أماكن الذاكرة كبوابة إدخال . شكل (10-13) يبين الدائرة المقترحة لذلك . عندما تقوم شريحة المعالج بتنفيذ الأمر : $LDA \ BFFF$

فإن العنوان \overline{BFFF} سيوضع على مسار العنوانين (A0 إلى A15) وسيكون الخط \overline{MEMR} في هذه المرة فعالا ، لذلك فإنه في هذه الحالة سيكون خرج فاكك الشفرة أو المنتقي فعالا كما في شكل (10-13) مما يسبب نبضة تزامن لشريحة العازل Buffer والتي بسببها يقوم العازل بنقل المعلومة الموجودة على دخله وهي المعلومة القادمة من الخارج ويضعها على مسار البيانات حيث يقوم المعالج بنقل هذه المعلومة إلى مسجل التراكم . لاحظ أن العازل Buffer لابد وأن يكون من النوع ثلاثي المنطق .

بالنظر إلى دوائر الإدخال والإخراج باستخدام خريطة الذاكرة وباستخدام الأمرين IN, OUT نلاحظ مدى التناقض بينهما ، ففي أي من الطريقتين لابد وأن يكون هناك فاكك شفرة أو منتقي لرقم البوابة ، الاختلاف هو أنه في حالة استخدام خريطة الذاكرة فإننا نشفر مسار العنوانين بالكامل (16 خطأ) وذلك في حالة التشفير الكامل ، أما في حالة استخدام الأمرين IN, OUT فإننا نشفر الثمانية خطوط الأولى فقط من مسار العنوانين وذلك أيضا في حالة التشفير الكامل . هناك أيضا فرق أساسي وهو أننا مع طريقة خريطة الذاكرة نستخدم خطى التحكم \overline{MEMR} و \overline{IOR} أما مع الطريقة الأخرى فإننا نستخدم الخطين \overline{IOW} . فيما عدا ذلك فإن الماسك Latch أو العازل Buffer يستخدمان في كل من الطريقتين إما لمسك معلومة الخرج في حالة بوابة الإخراج أو لعزل مصدر المعلومة عن مسار البيانات في حالة بوابة الإدخال .

لقد رأينا في جميع الدوائر السابقة استخدام عملية التشفير الكاملة للبوابات في الحالتين سواء باستخدام الأمرين IN, OUT أو باستخدام خرائط الذاكرة ، ونعني بالتشفيـر الكامل أن جميع خطوط العنـاوـين (الثـمانـيـة أو الـسـتـة عـشـر عـلـى حـسـب الـطـرـيقـة الـمـسـتـخـدـمـة) تـسـتـخـدـم فـي عـلـيـة التـشـفـير ، فـمـع خـرـائـط الـذـاـكـرـة مـثـلاـ استـخـدـمـنـا جـمـيع خـطـوـط مـسـار العـنـاوـين (A0 إـلـى A15) وـفـي حـالـة الـأـمـرـيـن IN, OUT استـخـدـمـنـا إـلـى 8 خـطـوـط الـأـولـى مـن مـسـار العـنـاوـين . فـي الـحـقـيقـة فـإـن دـائـرـة التـشـفـير مـن الـمـمـكـن أـن تـكـون أـبـسـط مـن الدـوـائـر السـابـقـة بـكـثـير لـو أـنـتـا اـسـتـخـدـمـنـا عـدـدـاـ أـقـلـ مـن خـطـوـط العـنـاوـين .



شكل (10-14) يـبيـن عـلـيـة تـشـفـير لـبـوـاـبـة إـخـرـاج باـسـتـعـمـال الـأـمـر OUT وـباـسـتـخـدـام خـطـ وـاحـدـ مـن مـسـار العـنـاوـين وـهـو الـخـط A0 . المشـكـلـة مـع مـثـلـ هـذـه الـبـوـاـبـة هـي أـن

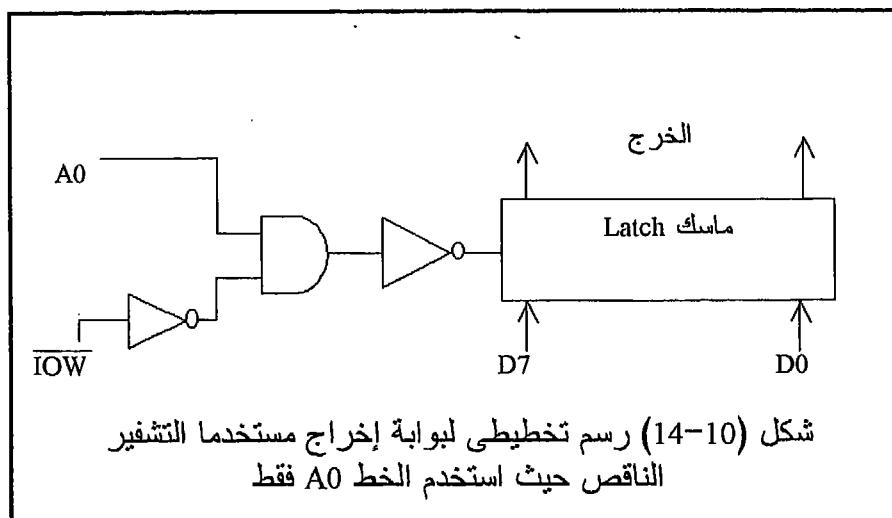
أى رقم أو عنوان تكون فيه البت A0 تساوى واحدا سيسبب فى تشغيلها . فمثلا جميع الأوامر التالية تصلح لتشغيل هذه البوابة :

OUT 01

OUT 03

OUT FF

وهكذا فإن أى رقم يبدأ بواحد في البت A0 سيشغل هذه البوابة . فى الحقيقة فإن بناء مثل هذه البوابات الناقصة التشفير ليس به أى عيب طالما أنه ليست هناك بوابات تحمل هذه الأرقام المتكررة ويجب أن يراعى ذلك فى عملية التصميم . نفس الكلام يمكن تطبيقه على عمليات تشفير البوابات التي تستخدم خرائط الذاكرة .



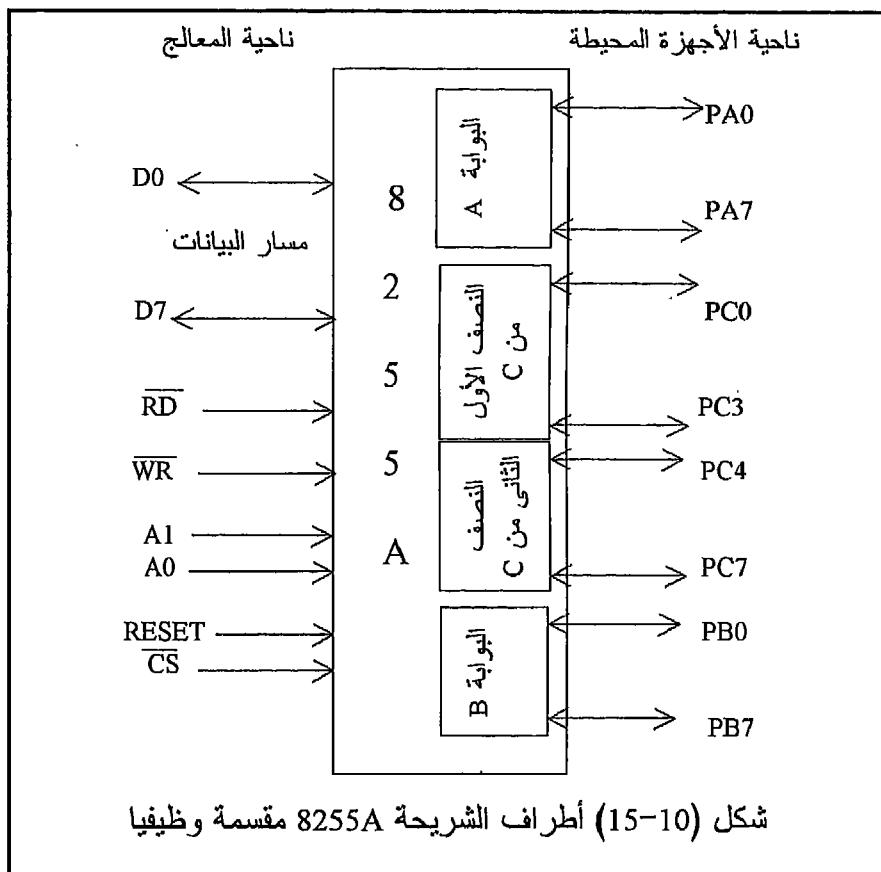
5-5 البوابات القابلة للبرمجة Programmable Peripheral Interface, PPI

الشريحة Intell 8255A هي إحدى الشرائح المهمة التي تصاحب دائما وفى الكثير من التطبيقات شرائح المعالج . إن هذه الشريحة تسمى Programmable Peripheral Interface واختصارا تكتب PPI . تحتوى هذه الشريحة كما سترى على ثلث بوابات كل منها ثمانى بิตات وهذه البوابات قابلة للبرمجة بمعنى أنه من داخل البرنامج يمكن جعل أى واحدة من هذه البوابات بوابة إدخال أو إخراج عن طريق أمرين فقط من أوامر لغة الأسsemblى وهذه تعتبر من المميزات العظيمة

لهذه الشريحة حيث أنها توفر الكثير من الدوائر التي كانت تستخدم في عملية تشفير البوابات ، فلنك أن تخيل أن أمامك ثلاثة بوابات لك الحرية في أن تجعل أي واحدة منهم إدخال أو إخراج كما تشاء ومن داخل البرنامج ودون اللجوء إلى أي تغييرات أو تعديلات في الدوائر التي تم بناؤها .

8255A تركيب الشريحة

هذه الشريحة يمكن تقسيمها إلى جزأين رئيسيين ، جزء يواجه شريحة المعالج والجزء الآخر يواجه العالم الخارجي . شكل (10-15) يبين رسمياً تخطيطياً لهذه الشريحة محتويها هذين الجزأين . الجزء المقابل للميكروبروسيسور يحتوى على الخطوط التالية :



شكل (10-15) أطراف الشريحة 8255A مقسمة وظيفياً

- خطوط مسار البيانات وعددتها 8 خطوط خطان
- خطوط مسار العنوانين وعددتها خطان

• خطوط مسار التحكم و عددها 4 خطوط (\overline{RD} , \overline{WR} , \overline{CS} , reset)

أما الجزء المقابل للعالم الخارجي فيحتوى الخطوط التالية :

- خطوط البوابة A و عددها 8 خطوط
- خطوط البوابة B و عددها 8 خطوط
- خطوط البوابة C و عددها 8 خطوط (4 خطوط + 4 خطوط)

بإضافة خطى القدرة (Vcc والأرضي) يصبح عدد خطوط أو أرجل هذه الشريحة 40 رجلا . لاحظ أن البوابة C لها خاصية منفردة عن البوابتين A, B وذلك لأنها يمكن برمجتها كبوابة 8 بتات أو كبوابتين كل منها 4 بتات أو أنها تستخدم خطوط تحكم للبوابتين B, A كما أنه يمكن عمل Set أو Reset لأى طرف من أطراف هذه البوابة بالذات كما سنرى فيما بعد . شكل (10-16) يبين الرسم الطرفي لهذه الشريحة .

PA3	1	40	PA4
PA2	2	39	PA5
PA1	3	38	PA6
PA0	4	37	PA7
\overline{RD}	5	36	\overline{WR}
\overline{CS}	6	35	RESET
GND	7	34	D0
A1	8	33	D1
A0	9	32	D2
PC7	10	31	D3
PC6	11	30	D4
PC5	12	29	D5
PC4	13	28	D6
PC0	14	27	D7
PCI	15	26	Vcc
PC2	16	25	PB7
PC3	17	24	PB6
PB0	18	23	PB5
PB1	19	22	PB4
PB2	20	21	PB3

8255A

شكل (10-16) أطراف الشريحة 8255A

من وجهة نظر المعالج فإن الشريحة 8255 تحتوى على أربعة مسجلات أو أربع بوابات أو أربعة أماكن يمكن عنونتها بعنوان خاص لكل منها وهذا هو السبب فى

أن الشريحة 8255 لها خطان فقط للعنوانين . ثلاثة من هذه المسجلات أو الأماكن هي المسجلات أو البوابات C, B, A وهي التي تستخدم لإدخال أو إخراج المعلومات وهي الثلاث بوابات التي في شكل (10-15) ، أما المسجل الرابع فهو مسجل تحكم Control register والذى عن طريقه يتم التحكم فى المسجلات C, B, A لجعلها إما بوابات إدخال أو إخراج والتحكم أيضاً فى طريقة تشغيل الشريحة A ككل .

العملية					
A1	A0	RD	WR	CS	
0	0	0	1	0	قراءة من البوابة A
0	1	0	1	0	قراءة من البوابة B
1	0	0	1	0	قراءة من البوابة C
0	0	1	0	0	كتابة في البوابة A
0	1	1	0	0	كتابة في البوابة B
1	0	1	0	0	كتابة في البوابة C
1	1	1	0	0	كتابة في مسجل التحكم
x	x	x	x	1	الشريحة غير فعالة
1	1	0	1	0	حالة غير ممكناً
x	x	1	1	0	الشريحة غير فعالة

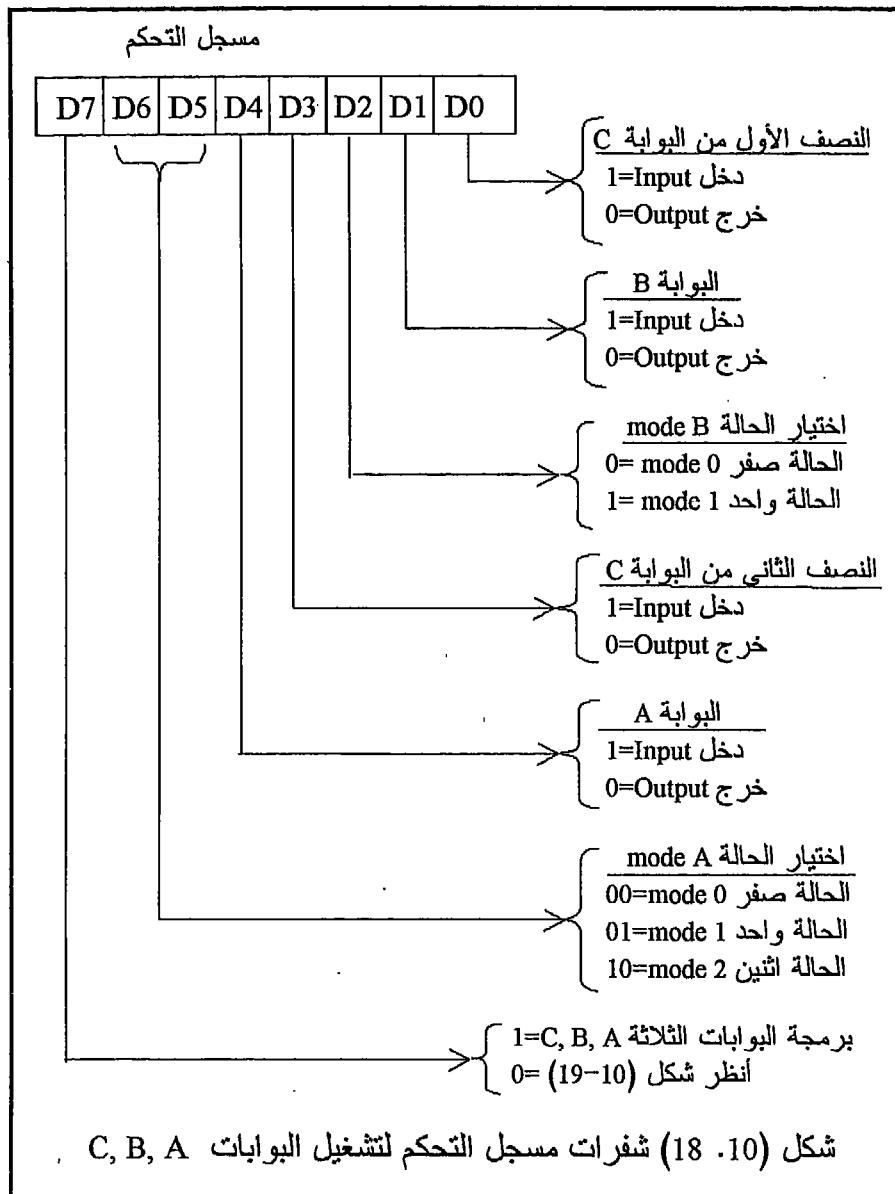
شكل (10-17) عنونة المسجلات في الشريحة 8255A . (x تعنى لا يهم)

شكل (10-17) يبين كيفية عنونة كل واحد من المسجلات A, B, C وكذلك مسجل التحكم وكيفية القراءة منها أو الكتابة فيها . فمثلاً لكي نرسل معلومة إلى المسجل A فإننا نضع الشفرة 00 على الخطين A1, A0 ونجعل خط الكتابة WR فعالاً بجعله يساوى صفرًا وكذلك الخط CS وهو خط اختيار الشريحة لابد وأن يكون أيضاً فعالاً بجعله يساوى صفرًا . تتبع طرق القراءة والكتابة في كل واحد من هذه المسجلات في شكل (10-17) . لاحظ أن مسجل التحكم يمكن الكتابة فيه فقط ولا يمكن قراءته حيث أن وظيفته لا تتطلب قراءته ، لذلك فإنه يسمى مسجل كتابة فقط write only register .

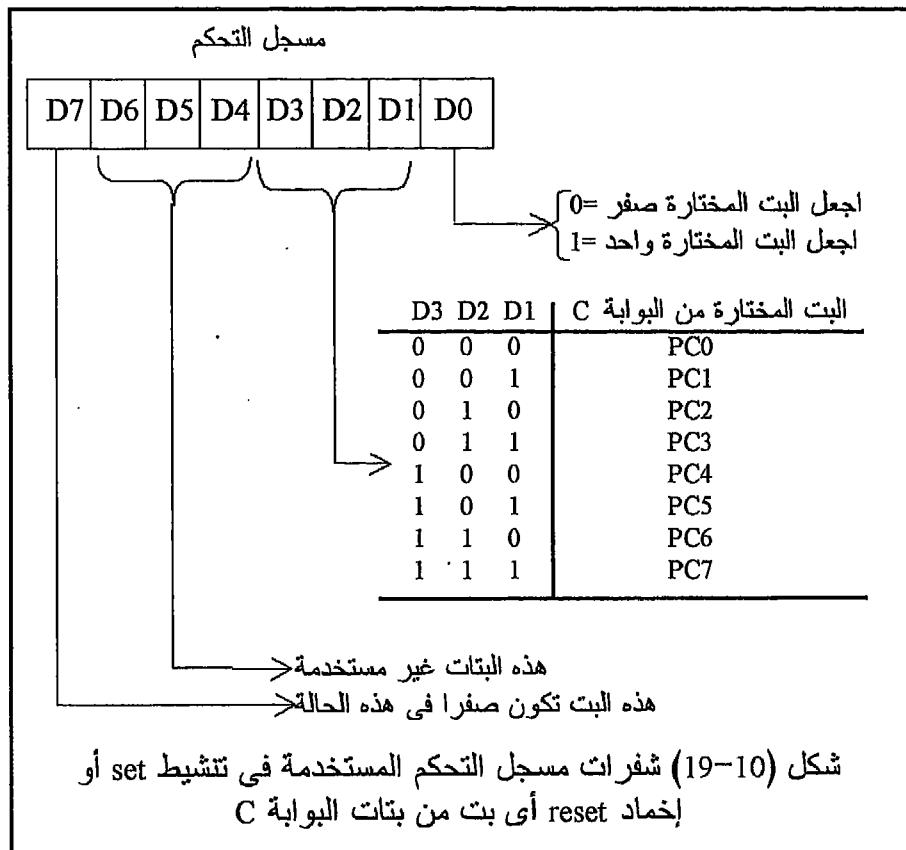
10-5-2 برمجة الشريحة 8255A

إن برمجة الشريحة 8255A لتعمل في أي وضع من أوضاع تشغيلها يتم عن طريق كتابة شفرة من ثماني بتات في مسجل التحكم ، وعلى ضوء هذه الشفرة تصبح أي واحدة من البوابات الثلاث في الوظيفة والحالة mode التي حددت لها

تبعاً لهذه الشفرة . تذكر هنا أن من الوظائف الخاصة بالبوابة C أنك يمكنك أن تتشط أو تخمد (reset أو set) أي طرف من أطرافها كما ذكرنا سابقاً . شكل (10-18) وشكل (19-19) يبيّنان وظيفة مسجل التحكم على ضوء الشفرة المخزنة فيه .



يهمنا هنا محتويات البت رقم 7 من مسجل التحكم فإذا كانت هذه البت تساوى واحداً فإن ذلك يعني أن باقى محتويات مسجل التحكم وهى البت رقم صفر إلى البت رقم 6 خاصة بتحديد الوظائف المختلفة للبوابات A, B, C على ضوء ما هو مبين في شكل (10-18). أما إذا كانت البت رقم 7 من مسجل التحكم تساوى صفرًا فإن ذلك يعني أن الشفارة الموجودة في باقى البتات خاصة بعمل set أو reset لأحد أطراف البوابة C على ضوء ما هو مبين في شكل (10-19) وتسمى حالة تشغيل/إخماد البت Bit Set Reset و اختصاراً تكتب BSR. وستتعارف عليها بالعربي هكذا (ب س ر).



مثال 1-10

ما هي الشفارة أو البايت التي نكتبها في مسجل التحكم للشريحة 8255A للحصول على الآتي :

البوابة B دخل ، والبوابة A خرج ، والبوابة C خرج ، والجميع يعمل فى mode 0 . سنفترض الحالة 0 فقط حاليا إلى أن يتم شرح باقى الحالات فى الجزء القادم .

بالنظر إلى شكل (10-18) نجد أنه لكي يكون النصف الأول من البوابة C خرجا فإن $D0=0$ ولكن تكون البوابة B دخلا فإن $D1=1$ ولكن تكون المجموعة B فى mode 0 فإن $D2=0$ ، بالنسبة للنصف الثانى من البوابة C فلکي يكون خرجا فإن $D3=0$ ولكن تكون البوابة A خرجا فإن $D4=0$ ولكن تعمل المجموعة A فى mode 0 فإن $D6D5=00$ ، وكما نعلم فإن $D7=1$ فى هذه الحالة . بذلك تصبح محتويات البايت المطلوبة هي :

D7 D6 D5 D4 D3 D2 D1 D0
1 0 0 0 0 0 1 0

بالنظام المستعشرى فإن هذه البايت تكون H 82H وال H تستخدم للدلالة على أن الرقم المجاور لها يكون فى النظام المستعشرى . المطلوب الآن هو كتابة أو إرسال هذا الرقم إلى مسجل التحكم فى الشريحة 8255A . لإرسال هذا الرقم إلى مسجل التحكم فى الشريحة 8255A فإنه كما نعلم أن خطى العناوين A1, A0 للشريحة 8255A موصلان بخطى العناوين A1, A0 القادمين من شريحة المعالج ولقد رأينا في شكل (10-17) أن عنوان مسجل التحكم هو 11 A1A0=11 لذلك فإنه لكي نرسل الرقم المطلوب إلى مسجل التحكم يكفي أن نكتب الأمرين التاليين :

MVI A,82H.

OUT 03

حيث الأمر الأول سيوضع الرقم 82H فى مسجل التراكم A والأمر الثانى سيخرج محتويات مسجل التراكم إلى البوابة رقم 3 وهى مسجل التحكم فى الشريحة 8255A لأن الرقم 3 سيجعل خطى العناوين A1, A0 يساويان 11 وهذا هو عنوان مسجل التحكم .

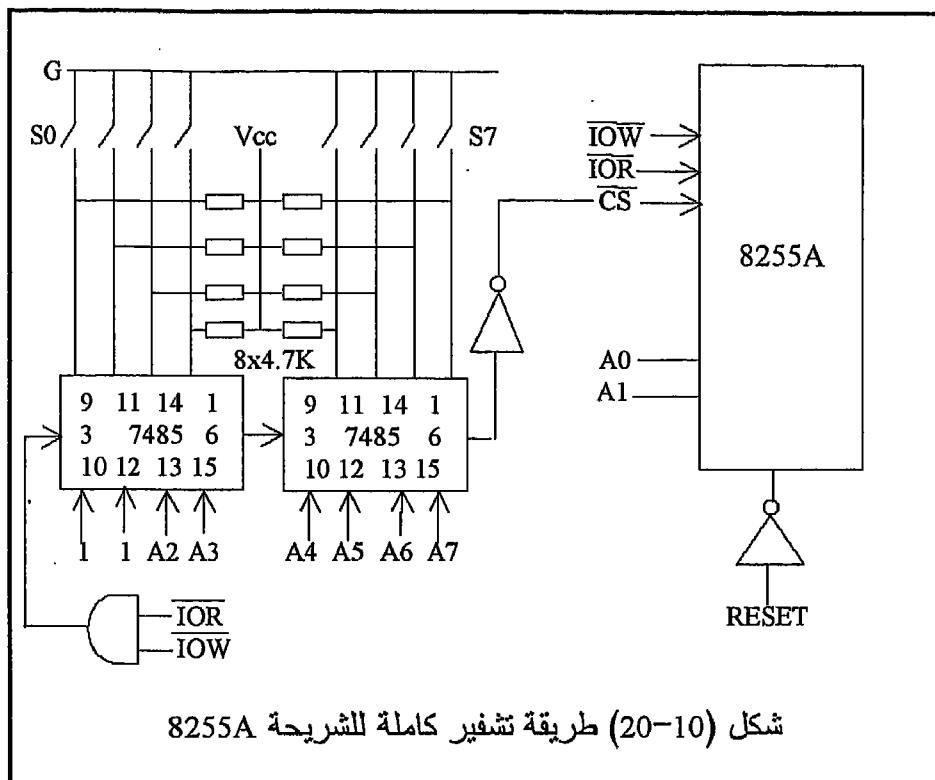
بذلك يمكن التعامل مع البوابات A, B, C حسب حالة كل منهم التي تم تحديدها فى مسجل التحكم بالرقم 82H . فمثلا الأمر 00 OUT 00 سيخرج محتويات مسجل التراكم على البوابة A لأن عنوان البوابة A هو A1A0=00 . وأما الأمر IN 01 فسيقرأ محتويات البوابة B ويضعها فى مسجل التراكم لأن عنوان البوابة B هو A1A0=01 . وأما الأمر 02 OUT 02 فسيقوم بعملية إخراج على البوابة C حيث عنوان البوابة C هو A1A0=10 . لاحظ أن العناوين التي استخدمناها للبوابات الثلاث فى الأوامر السابقة كانت 00 و 01 و 02 و 03 وكلها أرقاما مستعشرية وليس بالضرورة أن تكون هذه هي الأرقام التي يجب أن تستخدم فقط فى جميع الأحوال . النظرية هنا هي أن العنوان الذى نستخدمه يجب أن يحقق الشفرة المطلوبة لكل بوابة على الخطين A1, A0 . شكل (10-20) يبين الشريحة 8255A وقد استخدمت مع عملية تشفير كاملة للثمانية خطوط الأولى من مسار

العناوين . لاحظ عدم استخدام الخطين A1, A0 فى عملية التشفير ولكنها يوصلان مباشرة إلى الشريحة 8255A .

من شكل (10-20) نتائج العنوانين التالية للمسجلات الأربع :

	العنوان سبعى							
	A7	A6	A5	A4	A3	A2	A1	A0
0E	0	0	0	0	1	1	0	0
0D	0	0	0	0	1	1	0	1
0E	0	0	0	0	1	1	1	0
0F	0	0	0	0	1	1	1	1

مسجل التحكم CR



شكل (10-20) طريقة تشفير كاملة للشريحة 8255A

3-5-10 Modes تشغيل الشريحة 8255A

الشريحة 8255A لها ثلاثة حالات modes لتشغيلها وهي كالتالى :

Mode zero 1-3-5-10 الحالة صفر

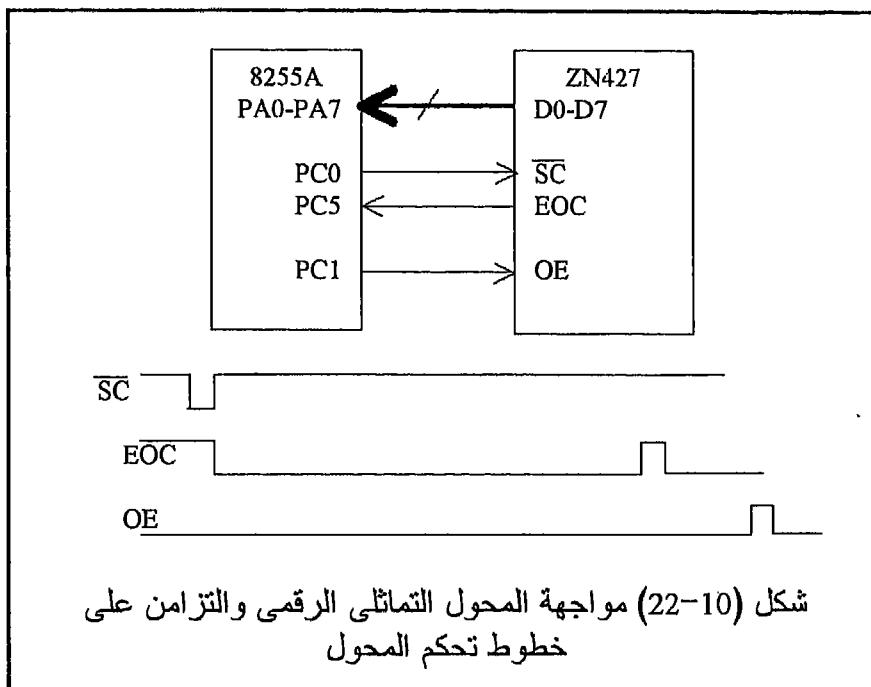
تتميز هذه الحالة بعمليات الإخراج والإدخال البسيطة التي لا تتطلب حواراً أو نظام مصافحة handshaking بين الشريحة والمعالج ، ففي هذه الحالة يمكن برمجة كل من البوابات A أو B أو C لتكون بوابة إخراج أو إدخال في أي مكان في البرنامج وعن طريق أمرين اثنين فقط كما سنرى بعد قليل . لاحظ أنه في هذه الحالة يمكن برمجة البوابة C كبوابة ثمانى بتات أو بوابتين كل منها أربع بتات . جميع هذه البوابات تكون فيها خاصية المسك latch للمعلومة عندما تعمل كبوابات إخراج ، وأما عندما تعمل كبوابات إدخال فلا تكون فيها هذه الخاصية . باعتبار البوابة C كبوابتين كل منهما 4 بتات فإنه يمكن أن يكون لدينا 4 بوابات يمكن برمجتها إخراج أو إدخال على حسب الشفرة التي ترسل إلى مسجل التحكم على ضوء ما رأينا في شكل (18-10) . شكل (21-10) يبين جميع هذه الحالات والشفرة المستعشرية التي ترسل إلى مسجل التحكم للحصول على كل حالة .

الشفرة المستعشرية إلى مسجل التحكم	البوابة A	البوابة C PC4-PC7	البوابة B	البوابة C PC0-PC3
80	إخراج	إخراج	إخراج	إخراج
81	إخراج	إخراج	إخراج	إدخال
82	إخراج	إخراج	إدخال	إخراج
83	إخراج	إخراج	إدخال	إدخال
88	إدخال	إدخال	إخراج	إخراج
89	إدخال	إدخال	إخراج	إدخال
8A	إدخال	إدخال	إدخال	إخراج
8B	إدخال	إدخال	إدخال	إدخال
90	إدخال	إخراج	إخراج	إخراج
91	إدخال	إخراج	إخراج	إدخال
92	إدخال	إخراج	إدخال	إخراج
93	إدخال	إخراج	إدخال	إدخال
98	إدخال	إدخال	إخراج	إخراج
99	إدخال	إدخال	إخراج	إدخال
9A	إدخال	إدخال	إدخال	إخراج
9B	إدخال	إدخال	إدخال	إدخال

شكل (10-21) جميع حالات الإدخال والإخراج للبوابات A و B و C والشفرة المستعشرية لكل حالة .

مثال 2-10

شكل (10-22) يبين رسمًا مصدوقياً لدائرة مواجهة مع المحول التماثلي الرقمي ZN427 مستخدماً الشريحة 8255A في الحالة 0 . المحول ZN427 له ثلاثة خطوط تحكم ، أحدها هو خط بداية التحويل SC Start Conversion; يجب أن يكون صفرًا لفترة زمنية وجيزة بحيث يبدأ المحول عملية التحويل عند الحافة الصاعدة لهذه الإشارة . بعد أن ينتهي المحول من عملية التحويل فإنه يعطى نبضة واحدة على خط نهاية التحويل EOC End Of Conversion; ولكن لا يعطى البيانات على خطوط الخرج الثمانية إلا إذا تم إعطاؤه نبضة واحدة على خط تنشيط الخرج Output Enable; OE الذي يقوم بتشييط البوابات الثلاثية الموجودة في خرج المحول . شكل (10-22) يبين أيضًا التزامن الموجود بين هذه الإشارات الثلاثة . لقد تم توصيل خرج المحول على البوابة A للشريحة 8255A ، أي أن البوابة A ستعمل كبوابة إدخال يقوم المعالج من خلالها بقراءة خرج المحول .



شكل (10-22) مواجهة المحول التماثلي الرقمي والتزامن على خطوط تحكم المحول

مطلوب أيضاً من المعالج أن يعطي للمحول نبضة بدأ التحويل SC وسيكون ذلك من خلال الخط رقم 0 في البوابة C وسيكون عن طريق استخدام الحالة (ب سر) التي سنستخدمها لتنشيط الخط PC0 كمارأينا في شكل (10-19) . بعد أن

يعطى المعالج نبضة بداية التحويل يبدأ في مراقبة خط نهاية التحويل القادم من المحول إلى الخط PC5 بحيث عندما يجد المعالج أن هذا الخط صعد إلى الواحد يفهم من ذلك أن المحول انتهى من عملية التحويل فيعطي له إشارة تنشيط الخرج بجعل الخط OE الموصول على PC1 يساوى واحد وذلك أيضاً باستخدام طريقة ال (ب س ر) . لذلك فإن النصف الأول من البوابة C سيعمل كبوابة إخراج وأما النصف الثاني فسيعمل كبوابة إدخال . البرنامج التالي يمكن استخدامه لقراءة خرج مثل هذا المحول . ملاحظة مهمة يجب أن نذكرها من هذا البرنامج وهي أن إرسال أي شفرة إلى مسجل التحكم لتنشيط أو إخماد أي بث من بثات البوابة C لا يؤثر على الإطلاق على الحالة التي عليها كل من البوابة A أو B أو C ، أي أن كل واحدة منها تبقى على الحالة التي عليها سواء كانت إدخالاً فستبقى إدخالاً أو كانت إخراجاً فستبقى كذلك أيضاً .

MVI A,98H ;	هذا الرقم يجعل PC0-PC3 إخراج والبوابة B إخراج
; غير مستخدمة في هذا المثال) و PC4-PC7 إدخال والبوابة A إدخال	(B)
;	راجع شكل (10-18).
OUT 03H ;	إخراج الرقم 98H إلى مسجل التحكم
START: MVI A,01	
;	تنشيط الخط PC0 عن طريق ب س ر راجع شكل (10.19)
OUT 03H ;	إخراج إلى مسجل التحكم
MVI A,00 ;	عمل إخماد للخط
OUT 03H ;	إخراج إلى مسجل التحكم
MVI A,01 ;	عمل ست للخط
OUT 03H ;	بذلك يكون الخط نزل من 1 إلى 0 ثم صعد وبذلك تم نبضة SC.
xx: IN 10 ;	القراءة البوابة C
ANA 20H ;	هذا الاختبار الخط PC5 لمعرفة إذا كان واحد أم صفر
JZ xx ;	قفز إلى xx لمعاودة القراءة طالما أن PC5 يساوى 0 .
MVI A,03H ;	تنشيط للخط PC1 لتنشيط الخط OE .
OUT 03H ;	إخراج إلى مسجل التحكم
MVI A,02H ;	إرجاع الخط OE=PC1 إلى الصفر ثانية .
OUT 03H ;	إخراج إلى مسجل التحكم
IN 00 ;	القراءة البوابة A
JMP START ;	لإعطاء نبضة بدأ تحويل جديدة

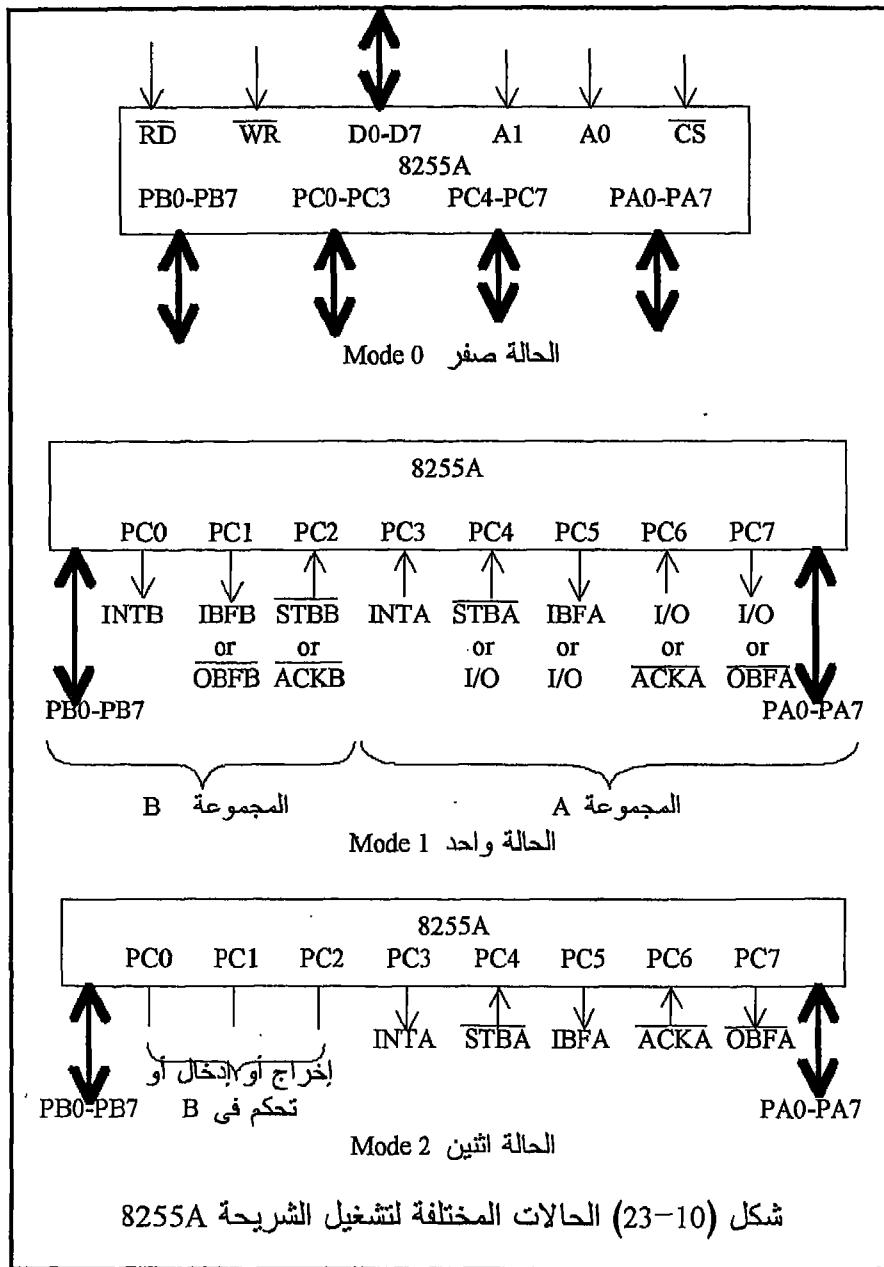
Mode one 10-5-2 الحالة واحد

في هذه الحالة يتم الإدخال أو الإخراج عن طريق البوابتين A و B فقط ويتم ذلك عن طريق نظام مصافحة Handshaking أو حوار بين الأجهزة المحيطة والمعالج من خلال الشريحة 8255A وتستخدم خطوط البوابة C في عمليات المصافحة هذه بحيث أن النصف الأول من خطوط البوابة C وهي PC0 إلى PC3 تستخدم كخطوط حوار تابعة للبوابة B ولذلك تسمى البوابة B والنصف الأول من البوابة C بالمجموعة B أو Group B . وأما النصف الثاني من خطوط البوابة C فيستخدم كخطوط مصافحة أو حوار تابعة للبوابة A ولذلك تسمى البوابة A والخطوط PC4 إلى PC7 بالمجموعة A . لاحظ أن البوابة C بأكملها لا يمكن استخدامها هنا في الإخراج أو الإدخال ولكن يمكن استخدام بعض خطوطها أحياناً كما في شكل (10-23) . تتميز هذه الحالة بأن البيانات سواء الدخلة أو الخارجة من البوابتين A أو B تمسك latched على هذه البوابات ، كما أن هذه الحالة توفر للمستخدم خطا يمكن منه مقاطعة المعالج . عمليات المقاطعة ستدرسها بالتفصيل في فصل خاص بذلك . شكل (10-23) يبين وظيفة كل خط من خطوط البوابة C كخطوط مصافحة لكل من البوابتين A و B في الحالة 1 وذلك عندما تستخدم كل من A و B كبوابة إدخال أو إخراج . شكل (10-24) يبين وظيفة خطوط البوابة C في حالة كون كل من البوابتين A و B بوابات إدخال فقط ويبين أيضاً التزامن بين هذه الإشارات في هذه الحالة . لاحظ أن البوابة B تستخدم الخطوط PC0, PC1, PC2 كخطوط مصافحة بينما تستخدم البوابة A الخطوط PC3, PC4, PC5 كخطوط مصافحة ويتبقي خطان وهما PC6, PC7 يستخدمان كخطوط إدخال أو إخراج على حسب البت D3 في مسجل التحكم . تتم عملية المصافحة مع الأجهزة المحيطة كما يلى :

- يقوم الجهاز الخارجي بوضع البيانات على البوابة التي يتعامل معها ولتكن البوابة A مثلاً ثم يضع صفراء على الخط STBA ليخبر الشريحة 8255A أنه قد وضع بait على البوابة A .

- ترد الشريحة 8255A على الجهاز الخارجي بأنها استقبلت المعلومة وتم مسکها على البوابة A عن طريق جعل الخط IBFA يساوى واحداً . عندما يرى الجهاز الخارجي أن الخط $IBFA=1$ يقوم بإرجاع الخط STBA إلى الواحدة $STBA=0$ إلى الصفر ثانية إلا إذا تم قراءة المعلومة بواسطة المعالج وذلك عند الحافة الصاعدة للإشارة RD الموصولة بالشريحة 8255A طرف 5.
- تقوم الشريحة 8255A بتحويل إشارة مقاطعة INTRA يمكن بها مقاطعة المعالج في حالة الرغبة في ذلك وذلك بتوصيلها إلى أي خط من خطوط المقاطعة على المعالج . لاحظ أيضاً من التزامن في شكل (10-24) أن هذا

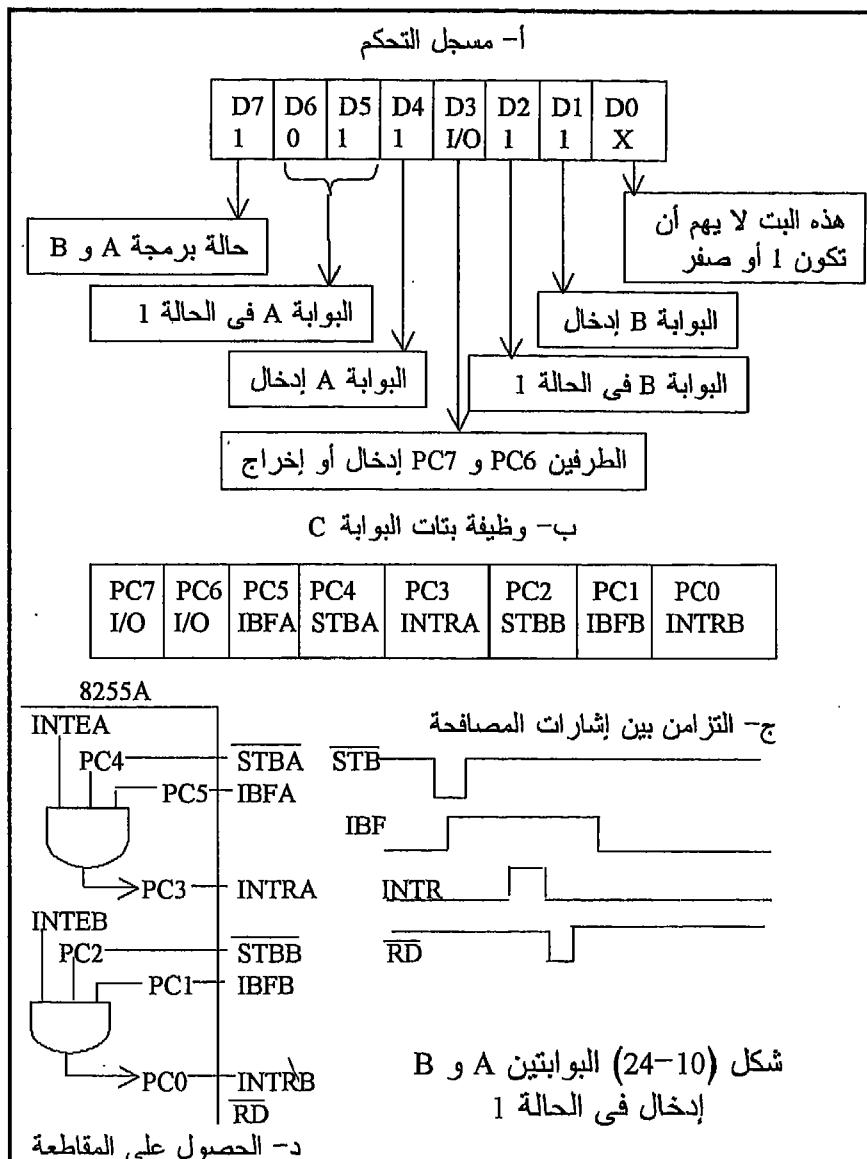
الخط يكون واحداً إذا كان الخط $\overline{STBA}=1$ والخط $IBF=1$ والبت $PC4=1$ في حالة التعامل مع البوابة A وأما في حالة البوابة B فاليت $PC2=1$. البتات $PC4, PC2$ يمكن جعلها واحداً باستخدام الـ بـ سـ رـ كما ذكرنا سابقاً وتذكر أن ذلك ليس له تأثير على وضع البوابات.



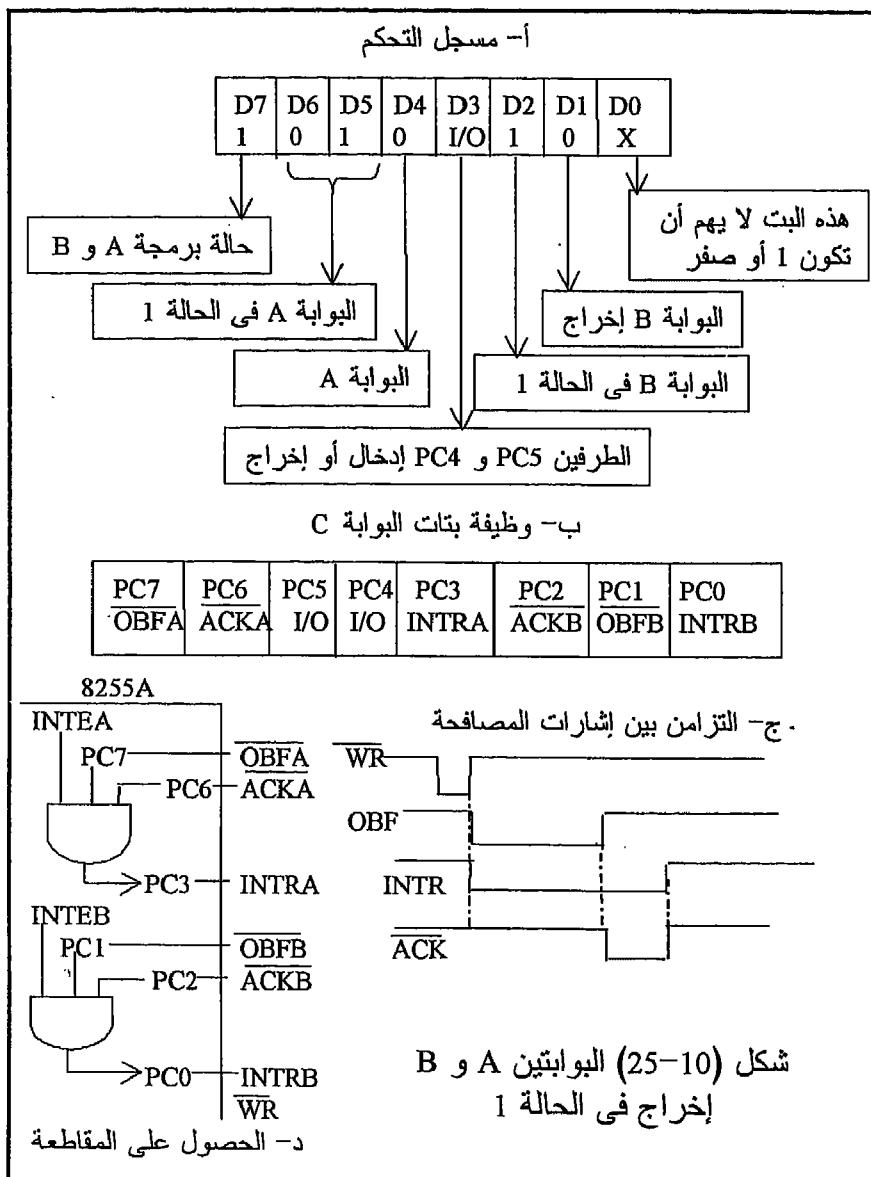
شكل (10-23) الحالات المختلفة لتشغيل الشريحة 8255A

شكل (25-10) يبين نفس الوظائف لخطوط البوابة C ولكن في حالة كون A و B بوابات إخراج ويبين أيضا التزامن بين هذه الإشارات . لاحظ أن البوابة A تستخدم الخطوط PC0, PC1, PC2 كخطوط مصافحة بينما تستخدم البوابة B الخطوط PC3, PC6, PC7 كخطوط مصافحة ويتبقي خطان وهما PC4, PC5 يستخدمان كخطوط إدخال أو إخراج على حسب البت D3 في مسجل التحكم .

تم عملية المصافحة كما يلى :



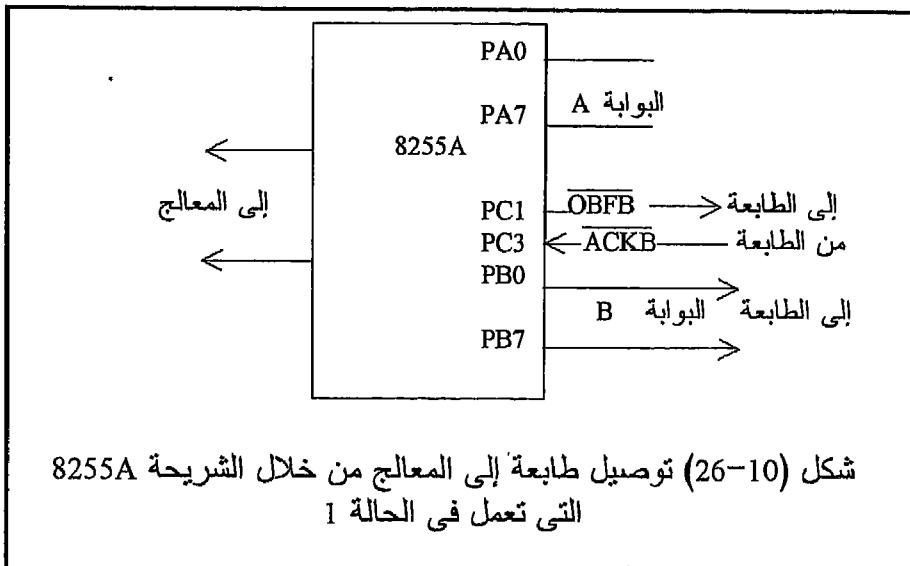
- عندما يقوم المعالج بكتابة أي بait إلى أي من البوابتين A أو B فإنه عند الحافة الصاعدة للإشارة WR تقوم الشريحة 8255A بجعل الخط OBFA يساوى صفرًا دلالة للمعالج أن هذه المعلومة قد تم مسكتها كما في التزامن الموضح في شكل (25-10) وعلى المعالج لا يرسل معلومات أخرى إلا بعد صعود هذا الخط للواحد مرة أخرى.



- عندما يرى الجهاز الخارجي أن الخط \overline{OBF} يساوى صفرًا يعرف أن هناك بait على البوابة وعليه قرأتها فيقوم بجعل الخط \overline{ACK} يساوى صفرًا وإرجاعه مرة أخرى للواحد لكي يخبر الشريحة 8255A أنه قدقرأ المعلومة . لاحظ من التزامن أن الحافة النازلة للخط \overline{ACK} تتسبب في إرجاع الخط \overline{OBF} إلى الواحد مرة أخرى .
- تقوم الشريحة 8255A بتمويل إشارة مقاطعة INTRA يمكن بها مقاطعة المعالج في حالة الرغبة في ذلك وذلك بتوصيلها إلى أي خط من خطوط المقاطعة على المعالج بغض النظر طلب بait جديد منه . لاحظ أيضاً من التزامن في شكل (10-25) أن هذا الخط ينزل إلى الصفر مع الحافة الصاعدة للخط \overline{WR} ويرجع إلى الواحد مرة ثانية عندما يكون كل من الخطين \overline{ACK} و \overline{OBF} يساوى واحداً والبت PC6 في حالة التعامل مع البوابة A والبت PC2 في حالة البوابة B تساوى واحداً أيضاً . البنات PC6, PC2 يمكن جعلها واحد باستخدام الـ B س ر كما ذكرنا سابقاً وتذكر أن ذلك ليس له تأثير على وضع البوابات .

مثال 3-10

شكل (26-10) يبين عملية توصيل طابعة مع المعالج من خلال البوابة B للشريحة 8255A وشكل (10-27) يبين البرنامج الذي يقرأ محتويات الذاكرة ابتداءً من المكان E100H وانتهاءً بالمكان E150H ويرسلها إلى الطابعة للطباعة . إن عملية تشفير البوابات الثلاث ، أي عنونتها ستتركها للقارئ يختار ما يشاء من عناوين للبوابات الثلاثة وسنفترض هنا أن هذه البوابات معروفة كالتالي :



00H	A البوابة
01H	B البوابة
02H	C البوابة
03H	مسجل التحكم

في هذا المثال تم استخدام البوابة B فقط وفي الحالة mode واحد وأما البوابة A وباقى البوابة C فيمكن استخدامها لأى أغراض أخرى . السؤال هو : ما هى الشفرة التي سنرسلها إلى مسجل التحكم لنجعل البوابة B إخراج وفي الحالة 1 وأما البوابة A وباقى البوابة C فسنفترضها إخراج فى الحالة 0 وذلك مع العلم أنها لن تستخدم ؟ الإجابة هي أن هذه الشفرة ستكون كالتالى كما تعلمنا من شكل (18-10) :

D7 D6 D5 D4 D3 D2 D1 D0
1 0 0 0 1 0 0 = 84H

- النصف الأول من البوابة C لا يهم أن يكون إخراجا أو إدخالا وبفرضه إخراجا تم وضع $D0=0$.
 - بما أن البوابة B ستكون إخراجا لذلك تم وضع $D1=0$.
 - بما أن B ستعمل في الحالة 1 لذلك تم وضع $D2=1$.
 - النصف الثاني من البوابة C لا يهم أن يكون إدخالا أو إخراجا وبفرضه إخراجا تم وضع $D3=0$.
 - بما أن البوابة A فرضت إخراجا لذلك فإن $D4=0$.
 - بما أن البوابة A فرضت في الحالة 0 لذلك كان $D6D5 = 00$.
 - بما أن هذه الشفرة تستخدم لتوظيف البوابات فإن $D7=1$.
- بالنظر إلى شكل (10-27) نجد أن البرنامج بدأ بإرسال الشفرة 84H كما سبق إلى مسجل التحكم ثم استخدم المسجلين HL كمؤشر إلى الحرف الجاهز للطباعة والمسجل B كعداد تنازلي للأحرف التي سيتم طباعتها بحيث تتف عمليه الطباعة عندما يصل هذا العداد إلى الصفر لأن البرنامج يتضمن هذا العداد بمقدار واحد كلما تمت طباعة حرف . تبدأ حلقة الطباعة من العلامة NEXT والتي عندها يتم قراءة البوابة C وحجب جميع بิตاتها بالقيمة 02H لمعرفة إذا كان الخط $\overline{OBF}=1$ يساوى صفرًا أم لا ، لأن صفرًا على هذا الخط معناه أن هناك حرف مازال ممسوكة على البوابة في انتظار الطابعة لاستلامه . بمجرد أن يصبح الخط $\overline{OBF}=1$ أي غير فعال يخرج المعالج من هذه الحلقة ويقوم بإرسال حرف جديد إلى الطابعة والدخول في نفس حلقة قراءة البوابة C مرة أخرى . تكرر هذه العملية إلى أن ينتهي المعالج من جميع الأحرف المطلوب طباعتها عندما يصل المسجل B إلى الصفر .

Mode two - 3-5-3 الحالة اثنان

البوابة A فقط هي التي يمكنها أن تستخدم في هذه الحالة وأما البوابة B ف تكون إما في الحالة صفر mode 0 أو في الحالة واحد mode 1 . عندما تشغّل البوابة A في هذا الحالة فإنها تسلك مسلك مسار بيانات بمعنى أنها تكون بوابة إخراج إذا كانت المعلومات خارجة وتكون بوابة إدخال إذا كانت المعلومات داخلة وذلك دون استخدام أي أمر أو اللجوء إلى مسجل التحكم لعمل ذلك . لذلك فإن البوابة A عندما تكون في هذه الحالة فإنها تستخدم خمسا من خطوط البوابة C في نظام المصافحة أو الحوار وهذه الخطوط هي PC3 إلى PC7 . إن هذه الحالة هي أعقد الحالات التي تستخدم مع الشريحة 8255A وعادة يستخدم في حالات الاتصال بين حاسبين أو بين المعالج والأقراص الصلبة ولذلك سنكتفي بهذه الإشارة عن هذه الحالة .

MVI A,84H ;	البوابة B إخراج في الحالة 1 و A و C إخراج في الحالة 0 ;
OUT 03H ;	إرسال إلى مسجل التحكم
LXI H,E100 ;	إشارة لبداية الأحرف المطلوب طباعتها
MVI B,50H ;	المسجل B عداد لهذه الأحرف
NEXT: IN 02 ; PC1	قراءة البوابة C لمعرفة حالة الخط OBF وهو الخط PC1
ANI 02H ;	حجب لجميع البيانات ما عدا PC1
JZ NEXT ;	دوران في الحلقة طالما أن PC1=OBF=0
MOV A,M ;	إحضار حرف ووضعه في المركم
OUT 01H ;	إخراج على البوابة B (الطابعة)
INX H ;	إشارة إلى الحرف التالي
DCR B ;	إنفاص العداد بمقدار 1
JNZ NEXT ;	قفز لطباعة الحرف التالي

شكل (26-10) برنامج الطابعة الموصلة في شكل (27-10)

6-10 تمارين

1. ما المقصود بالإدخال والإخراج ؟
2. ما هو الفرق بين الإدخال والإخراج والكتابة والقراءة من الذاكرة ؟
3. أيهما أسرع ، إرسال واستقبال البيانات على التوالي ، أم على التوازي ؟ اذكر بعض التطبيقات التي تستخدم كل نوع ؟

4. عند تتنفيذ المعالج للأمر OUT 00 مثلا ، فإنه يقوم بالتأثير على المسارات الثلاثة ، اذكر هذه التأثيرات ؟
5. عند تتنفيذ المعالج للأمر IN 00 مثلا ، فإنه يقوم بالتأثير على المسارات الثلاثة ، اذكر هذه التأثيرات ؟
6. أعد رسم شكلي (10-7 و 10-8) مستخدما فاكك شفرة decoder بدلا من مقارن العناوين ؟
7. أعد رسم شكلي (10-9 و 10-10) مستخدما فاكك شفرة decoder بدلا من مقارن العناوين ؟
8. مطلوب توصيل بوابة إخراج واحدة فقط وأخرى إدخال على المعالج ، ارسم أبسط دائرة للتوصيل ؟
9. ما هو المقصود بالتشفير الكامل والتشفير الناقص ؟ من أى أنواع التشفير تكون الدائرة التى وصلتها فى السؤال الثامن ؟
10. عند تتنفيذ المعالج للأمر STA adr مثلا ، يقوم بالتأثير على المسارات الثلاثة ، اذكر هذه التأثيرات ؟ كيف يمكن استغلال هذه التأثيرات لبناء بوابة إخراج ؟
11. عند تتنفيذ المعالج للأمر LDA adr مثلا فإنه يقوم بالتأثير على المسارات الثلاثة ، اذكر هذه التأثيرات ؟ كيف يمكن استغلال هذه التأثيرات لبناء بوابة إدخال ؟
12. ماذا تعنى خريطة الذاكرة لأى ميكروكمبيوتر ؟ ارسم خريطة الذاكرة للميكروكمبيوتر الذى تستخدمه وادرسها جيدا ؟
13. أعد رسم شكلي (10-12 و 10-13) مستخدما مقارن عناوين بدلا من فاكك الشفرة decoder ؟
14. هل يمكن بناء بوابة إخراج تأخذ نفس عنوان أحد أماكن الذاكرة الموجودة فعلا ؟
15. هل يمكن بناء بوابة إدخال تأخذ نفس عنوان أحد أماكن الذاكرة الموجودة فعلا ؟
16. من وجهة نظر المعالج فإن الشريحة 8255A تتكون من 4 مسجلات يمكن القراءة منها والكتابة فيها ، هل هذه العبارة صحيحة أم خطأ ؟
17. هل من الضروري أن تكون عناوين البوابات A و B و C متتابعة أى 5A و 5B و 5C مثلا وذلك في الشريحة 8255A ؟
18. هل يمكن استخدام الشريحة 8255A و عنوانها بطريقة خرائط الذاكرة ؟ أم أنه لابد من استخدامها مع الأمرين IN و OUT ؟
19. اشرح مع التبسيط الحالات modes الثلاثة لتشغيل الشريحة 8255A ؟

الفصل الحادى عاشر

دراسة لباقي أطراف المعالج من خلال

تطبيق: التحكم فى إشارة مرور

Control of a Traffic Light

1-11 مقدمة

إنه في الكثير من الأحيان وعند ذكر التطبيقات التي تستخدم الميكروكمبيوتر للتحكم في أي عملية صناعية يتبرد إلى ذهنا فوراً الميكروكمبيوتر بصورته المركبة والمعقدة حيث الشاشة ولوحة المفاتيح والطابعة ووحدة التحكم المركزي CPU والكمية الهائلة من الذاكرة التي قد تصل إلى الكثير من الميجابايتات ، في حين أن العملية من الممكن أن تكون أبسط من ذلك بكثير كما سنرى . سناحول في هذا الفصل أن نقدم عرضاً مفصلاً وشرياً دقيقاً لدائرة المعالج على كارت الإلكتروني واحد one board وهذا الكارت يمكن استخدامه في الكثير من أغراض التحكم ومنها على سبيل المثال عملية التحكم في متغيرات تتك تغيير المياه التي ذكرناها في مقدمة الفصل العاشر وسوف نستخدم هذه الدائرة في هذا الفصل في عملية التحكم في إشارة مرور رباعية كتطبيق على ذلك .

2-11 تركيب الدائرة

إن الميكروكمبيوتر بمعناه العام والشامل وكما ذكرنا في الفصل الأول من هذا الكتاب يتركب من شاشة للعرض ولوحة مفاتيح ووحدة تحكم مركزي CPU وكمية من الذاكرة تقل أو تكثُر على حسب متغيرات وأوضاع كثيرة . إن الكثير من التطبيقات وبالذات التي تستخدم المعالج في أغراض التحكم لا تحتاج إلى كل هذه الإمكانيات ، فما فائدة شاشة العرض مثلاً في دائرة نريد تصميمها للتحكم في سرعة محرك والحفظ على ثباتها ؟ أو ما فائدة لوحة المفاتيح أو الكمية الهائلة من ال RAM في دائرة نريد تصميمها للتحكم مثلاً في دائرة مرور في تقاطع معين داخل مدينة ؟ من هنا كان السؤال عن ما هي أقل الإمكانيات التي نستطيع بها أن نصمم دائرة تكون سهلة البناء ، رخيصة التكاليف ، وتقوى بمعظم التطبيقات التي تحتاج إلى المعالج كعنصر أساسى في تطبيقات التحكم الآلى ؟

ت تكون أي دائرة تستخدم المعالج في أغراض التحكم العامة من الأجزاء التالية :

1. المعالج وقد تم تهيئه جميع مساراته لعملية المواجهة مع الأجهزة المحيطة
2. شريحة ذاكرة EPROM تحتوى البرنامج الذى سيقوم بالعملية التى تستخدم من أجلها دائرة المعالج .
3. شريحة RAM قد تكون هناك الحاجة إليها من قبل البرنامج السابق ، وإذا لم تكن هناك حاجة إليها يمكن في هذه الحالة الاستغناء عنها .
4. عدد من بوابات الإدخال والإخراج على حسب الحاجة والتطبيق الذى تستخدم من أجله الدائرة المذكورة .

لكى نفهم طريقة عمل هذه الدائرة فنحن نعلم أن المعالج عند إعطائه نبضة RESET أو عند بداية تشغيله يبدأ التنفيذ من عنوان معين وهذا العنوان يختلف من معالج لآخر ، فإذا جعلنا هذا العنوان هو أول عنوان فى البرنامج الذى كتبناه لهذا الغرض (غرض التحكم فى أى عملية صناعية) الموجود فى شريحة ال EPROM والتى تم توصيلها بحيث تعمل مع هذا العنوان ، فإنه بمجرد تشغيل المعالج سينفذ البرنامج ويعامل مع بوابات الإدخال والإخراج وعلى حسب ظروف العملية التى صمم من أجلها . المفروض طبعاً أن هذا البرنامج سيكون برماجاً يدور فى حلقة لا نهائية تجعل المعالج فى حالة تنفيذ مستمرة للبرنامج .

11-2-1 مثال توضيحي

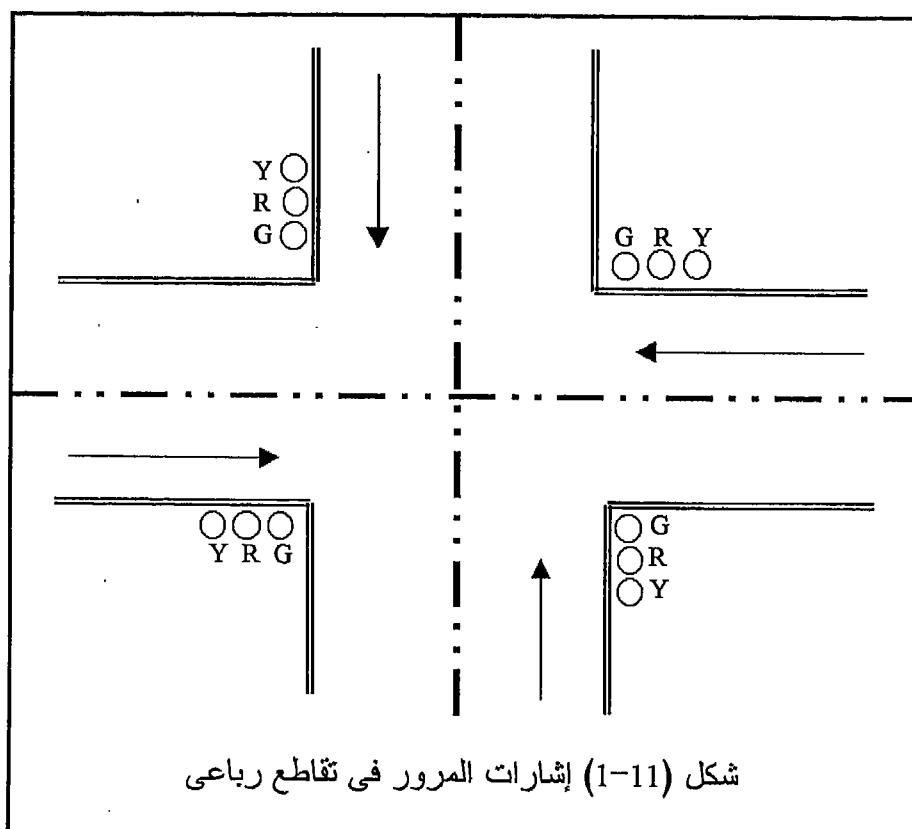
كمثال على ذلك سنقوم فى هذا الفصل إن شاء الله بعمل نظام تحكم فى إشارة مرور فى تقاطع رباعي كالما بين فى شكل (11-1) بحيث يحتوى كرت التحكم على مفتاح (خط تحكم يدوى يستعمل بواسطة رجل المرور عند اللزوم) ، بحيث عندما يكون هذا المفتاح واحداً فإن الإشارات تعمل فى الوضع الطبيعي وعندما يكون هذا المفتاح صفرًا فإن اللون الأصفر فى جميع الإتجاهات يضيء ويطفىء flashing بتردد معين ول يكن نصف ثانية .

سنقوم هنا ببناء الدائرة مستخدمين المعالج 8085 بعد تجربتها والتأكد من صحتها معملياً ، وسنترك للقارئ حرية إعادة بناء الدائرة مستخدماً إما المعالج Z80 أو أى معالج آخر إذا كان يفضل العمل بأحددها وسوف نشير إلى أى معلومات ضرورية لذلك فى حينها .

سيقوم المعالج لحل هذه المشكلة بادارة عدد 12 لمبة فى الأركان الأربع من التقاطع منها أربعة باللون الأخضر G, Green ، واحدة فى كل ركن ، وأربعة باللون الأصفر Y, Yellow ، واحدة فى كل ركن ، وأربعة باللون الأحمر R, Red . الإنارة هذا العدد من اللمبات (12) سنحتاج إلى بوابتي إخراج سنتستخدم منها 12 بت لللمبات والأربع برات الباقية سنتركها بدون استخدام حالياً أو لما قد يجد فى المستقبل من إضافات بالنسبة للدائرة . بالنسبة لمفتاح التحكم الذى سيشغل الإشارة إما فى الوضع الطبيعي أو الوضع الترددى flashing فإننا سنحتاج لبوابة إدخال يقرأ منها المعالج قيمة هذا المفتاح باستمرار قبل البدء فى أى دورة من دورات الإشارة كما سنرى بعد قليل . لذلك فإننا سنحتاج للشراطتين التالية لكى نتم عملية بناء دائرة التحكم فى إشارة المرور :

- شريحة المعالج Intel 8085 وقد وصلت جميع أطرافها إلى الجهد المناسب (سواء أرضى أو Vcc) وسنرى كيفية توصيل هذه الأطراف فى الجزء القادم .

- شريحتين 74374 لفصل buffer مسار العناوين وقد رأينا ذلك في الفصل الثامن .
- شريحة 74245 لفصل buffer مسار البيانات كما في الفصل الثامن أيضا .
- شريحة 74138 وشريحة 74125 للحصول على خطوط التحكم MEMR و IOW و IOR و MEMW وقد رأينا ذلك أيضا في الفصل الثامن .
- ثلاثة بوابات ، وهذه قد فضلنا أن نحصل عليها من الشريحة 8255A التي شرناها في الفصل العاشر .
- شريحة EPROM وهي الشريحة 2716 التي تحتوى على 2 كيلوبايت EPROM حيث سيتم حرق (كتابة) البرنامج عليها .
- بعض الشرائح البسيطة مثل 7408 وهى AND وشريحة عاكس التي تحتاجها عملية التشفير المبسطة للبوابات وشريحة الذاكرة .
- القليل من المقاومات والمكثفات كما سنرى بعد قليل في الدائرة الكاملة لهذا الكارت . سنطلق على هذه الدائرة إسم دائرة الميكروكمبيوتر ذي الكارت الواحد one board microcomputer



يجب على مستخدمي المعالج Z80 والمعالجات الأخرى أن يراعوا استخدام الشرائح المناسبة كما رأينا في الفصل الثامن عند فصل مسارات كل واحد من هذه المعالجات .

عند تفكيز هذا الكارت ستواجهنا بعض الأطراف لشريحة المعالج التي لا نعرف وظيفتها بالضبط لأننا لم ندرسها حتى الآن ولذلك فإننا لا نعلم ماذا نفعل بهذه الأطراف ، هل نتركها مفتوحة floating أم هل نوصلها بالأرضي أم Vcc . الجزء القائم من هذا الفصل سنشرح فيه وظيفة كل طرف من هذه الأطراف وماذا نفعل به على الكارت وذلك لكل واحد من المعالجين 8085 و Z80 وذلك قبل أن ندخل في تفاصيل حل مثل إشارات المرور .

8085 الأطراف الأخرى للمعالج

لقد درسنا في الأجزاء السابقة وظيفة أطراف المسارات الثلاثة للشريحة 8085 (40 طرفا) وهي كالتالي :

1. أطراف مساري العنوانين والبيانات وعددتها 16 خط .
2. أطراف التحكم وعددتها 3 خطوط (WR, RD, IO/M).
3. الطرف ALE .
4. طرفان للقدرة Vcc والأرضي .

مجموع هذه الأطراف هو 22 طرفا ويتبقي 18 طرفا لم نعلم عنها شيئا حتى الآن وسنقوم في هذا الجزء بشرح سريع لوظائف هذه الأطراف المتبقية . شكل (11-2) يبين إعادة لرسم أطراف الشريحة 8085 لتسهيل عملية المتابعة .

Clock signals اشارات التزامن

الشريحة 8085 تحتوى على مذبذب وهذا المذبذب يأخذ تردداته من بللورة أو كريستال crystal توصل بين الطرفين 1 و 2 للشريحة . هذا المذبذب ينتج عنه موجة جيبية ذات تردد يساوى 4 ميجا هرتز . الشريحة 8085 تحتاج إلى نبضات تزامن مربعة ذات تردد يساوى 2 ميجا هرتز ولذلك فإنه وكما هو موضح في شكل (11-3) فقد وصل خرج المذبذب الجيبى على مقارن من نوع سميت ليقوم بتحويل الموجة الجيبية إلى موجة مربعة ثم بعد ذلك أدخلت هذه الموجة المربعة على قاسم ليقوم بقسمة تردد الموجة المربعة على 2 فتحصل عند خرج القاسم على موجة مربعة ذات تردد 2 ميجا هرتز حيث تستخدم هذه الموجة في جميع أغراض التزامن والتشغيل داخل الشريحة 8085 . شكل (11-3) يبين كيفية الحصول على هذه النبضات . هناك الكثير من الشرائح المحيطة والمساعدة للشريحة 8085 والتي تحتاج إلى نفس نبضات التزامن التي تعمل عليها ، لذلك

فقد تم إخراج نبضات التزامن على الطرف 37 لشريحة المعالج لكي تتمكن الشرائح والأجهزة المحيطة من الإستفادة منها .

X1	1	40	Vcc
X2	2	39	HOLD
Reset out	3	38	HLDA
SOD	4	37	CLK OUT
SID	5	36	Reset in
TRAP	6	35	READY
RST7.5	7	34	IO/M
RST6.5	8	33	S1
RST5.5	9	32	RD
INTR	10	31	WR
INTA	11	30	ALE
AD0	12	29	S0
AD1	13	28	A15
AD2	14	27	A14
AD3	15	26	A13
AD4	16	25	A12
AD5	17	24	A11
AD6	18	23	A10
AD7	19	22	A9
Vss	20	21	A8

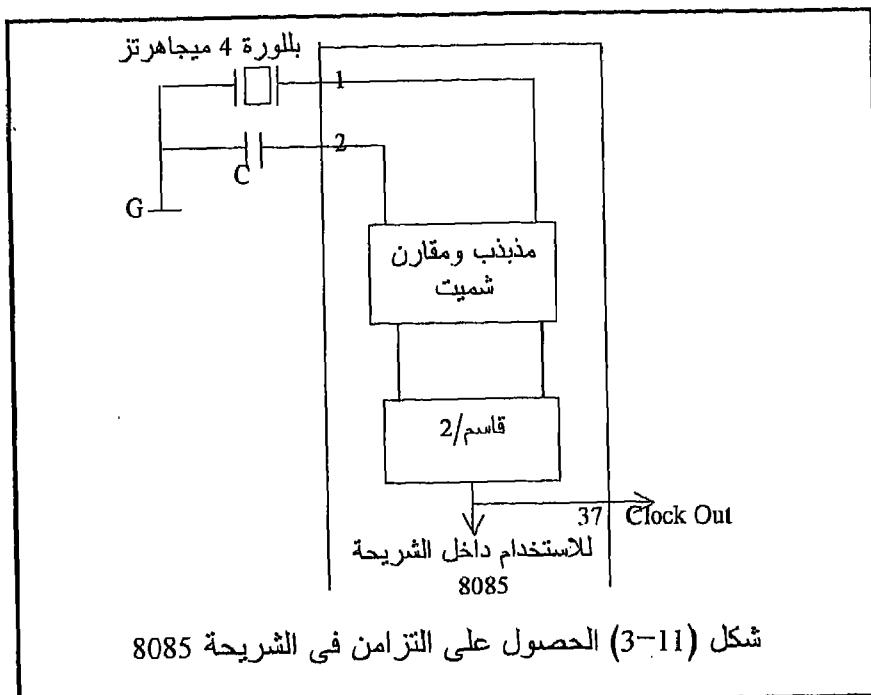
شكل (2-11) الشريحة 8085

11-3-2 بدأ وإعادة تشغيل المعالج

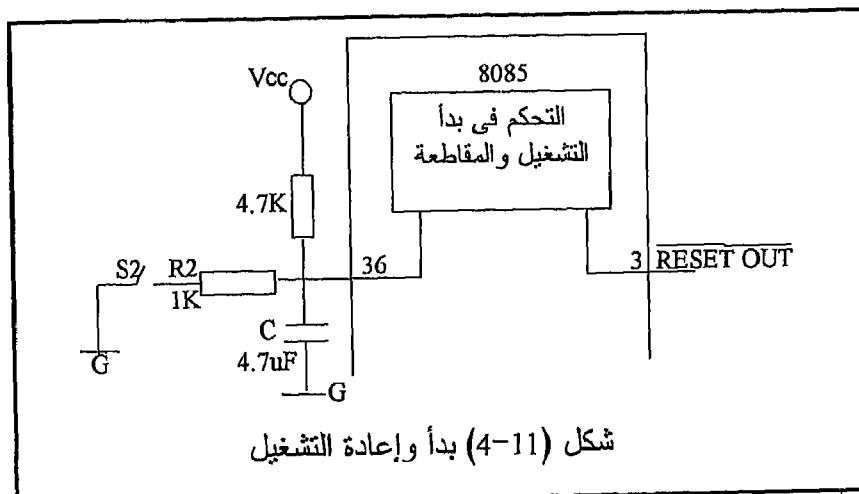
Starting and resetting the processor

عند وضع صفر على الطرف 36 للشريحة 8085 وهو الطرف RESET IN فإن عداد البرنامج داخل الشريحة يصبح 0000H (ستعشري) وعند عودة هذا الطرف إلى الواحد فإن المعالج يبدأ تنفيذ البرنامج الخاص بإعادة التشغيل وال موجود عند المكان 0000H في الذاكرة . شكل (2-11) يبين دائرة يمكن استخدامها في عملية بدأ وإعادة التشغيل . عند بدء التشغيل فإن المكثف C يشحن من خلال المقاومة R1 إلى أن يصل جهده إلى القيمة high أو الواحد الثنائي عندما تبدأ الشريحة في العمل من العنوان 0000H كما ذكرنا . إن هذا التأخير الناتج من شحن المكثف

يعتبر ضرورياً حتى نضمن عدم تشغيل الشريحة قبل أن يستقر الجهد وحتى تلتافي المشاكل اللحظية Transients التي تحدث عند بدأ التشغيل.



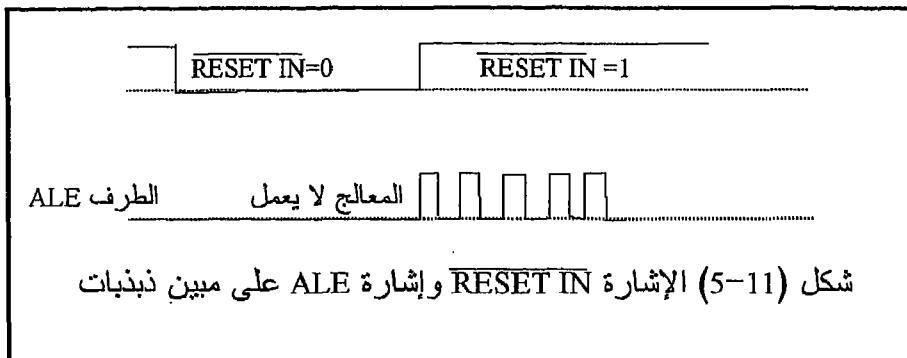
شكل (3-11) الحصول على التزامن في الشريحة 8085



شكل (4-11) بدأ وإعادة التشغيل

يمكن في أي وقت أثناء عمل الشريحة 8085 إرسال إشارة إعادة تشغيل إليها من خلال الطرف 36 عن طريق قفل المفتاح S2 حيث عندها يبدأ المكثف C في

التريغ من خلال المقاومة R2 وطالما أن المفتاح S2 مغلق فإن الشريحة 8085 لن تعمل . عند ترك المفتاح S2 يفتح يبدأ المكثف C في عملية الشحن من خلال المقاومة R1 حيث تبدأ خطوات إعادة التشغيل التي شرحناها سابقا . يمكن رؤية عملية بدء وإعادة التشغيل على مبين التذبذبات كما في شكل (5-11) .



كما نعلم فإن الطرف ALE في أثناء عمل الشريحة 8085 يجب أن تكون عليه موجة مربعة تعكس حالة الإشارة الموجودة على أطراف المسارات AD0-AD7 ، فإذا كان هذا الخط low فإن الإشارة تكون بيانات ، أما إذا كان هذا الخط high فإن الإشارة تكون عناوين وكما نعلم فإن المعالج أثناء عمله لابد وأن يكون في حالة تعامل مع عناوين أو بيانات لذلك فإننا يجب أن نرى موجة مربعة على الخط ALE وهذه يمكن استخدامها للدلالة على أن المعالج يعمل . شكل (5-11) يبين الإشارة الموجودة على الطرف ALE مع الإشارة الموجودة على الطرف RESET IN ، لاحظ في هذا الشكل أنه طالما أن طرف بدء التشغيل low فإنه ليس هناك أي إشارة على الطرف ALE وعند رجوع طرف بدء التشغيل إلى high تظهر الموجة المربعة على الطرف ALE دلالة على أن المعالج بدأ في العمل .

3-3 الطرفان HOLD و HLDA

لكي يتم اتصال بين أي جهاز خارجي والذاكرة فإن ذلك يكون عادة عن طريق المعالج ، حيث يأخذ المعالج المعلومة من الجهاز الخارجي من خلال مسار البيانات ويقوم بإرسالها إلى المكان المحدد في الذاكرة من خلال مسار البيانات وبمساعدة مساري العناوين والتحكم . هذه الطريقة من الاتصال تسمى بطريقـة الاتصال غير المباشر مع الذاكرة حيث يكون الاتصال عن طريق المعالج . هناك طريقة الاتصال المباشر DMA Direct Memory Access والتي يقوم فيها الجهاز الخارجي بإرسال معلوماته مباشرة إلى الذاكرة دون اللجوء إلى المعالج .

من أمثلة ذلك نسخ المعلومات من ذاكرة الحاسوب إلى الإسطوانة اللينة Flopy disk والعكس . في حالة الإتصال المباشر بالذاكرة فإن المعالج يجب عليه في هذه الحالة أن ينعزل أو يتراك أو ينفصل عن المسارات جميعها لاستخدامها الجهاز الخارجي في عملية الإتصال بالذاكرة وحتى لا يحدث أي تصادم في المعلومات على هذه المسارات . عندما يحتاج الجهاز الخارجي للمسارات ليقوم من خلالها بالإتصال المباشر بالذاكرة فإنه يخبر المعالج بذلك عن طريق اعطاء إشارة High على الخط HOLD . كلمة Hold تعنى أمسك أو قف أو تجمد على وضعك الحالى وهذا هو ما يحدث فعلاً للمعالج . عندما يتبع المعالج وجود high على الطرف HOLD وهو الطرف رقم 39 فإنه يقوم بانهاء دورة الماكينة Machine cycle الحالية والتي يقوم بتنفيذها ثم يوقف تنفيذ البرنامج ويوضع جميع خطوط المسارات في حالة المقاومة العالية وهي حالة العزل أو الانفصال ويتجدد على هذا الوضع إلى أن يعود الخط HOLD إلى الصفر مرة أخرى . بذلك يكون المعالج قد انفصل عن المسارات ، وعند ذلك فإنه يعطي إشارة للدالة على أنه انفصل عن المسارات عن طريق وضع الخط رقم 38 وهو HLDA في وضع الـ high أي واحد . HLDA تعنى الاعتراف بحالة التجمد Hold Acknowledge . المفروض على الجهاز الخارجي بعد أن يرسل الإشارة high إلى المعالج على خط الـ HOLD أن يتبع الإشارة الموجدة على الخط HLDA ، فعندما يصبح هذا الخط واحداً فإن ذلك يعني أن المعالج قد انفصل عن المسارات وعندها فقط يستطيع الجهاز الخارجي أن يستخدم المسارات . يمكن استخدام الخط HLDA أيضاً بحيث عندما يكون واحداً فإنه يضع جميع الشرائح المستخدمة في عملية فصل المسارات الثالثة في الحالة الثالثة وهي حالة المقاومة العالية . في دائرة الميكروكمبيوتر التي سنبنيها نريد أن يكون المعالج في حالة عمل مستمر ولكن نقاطعه أبداً أو نطلب منه مساراته ، ولذلك سنوصل خط الدخل HOLD بالحالة low أي غير فعال ، وخط الخرج HLDA سنتركه مفتوحاً ولأن نوصله بأى شيء لأنه يمثل إشارة خرج من المعالج وتركه مفتوحاً لن يؤثر على عمل المعالج .

11-3-4 الطرف READY

إن جميع أجهزة الميكروكمبيوتر تكون بها إمكانية تنفيذ البرامج بنظام الخطوة بخطوة Step by step execution في حالة تعاملها بلغة الأسsembl . في هذا النظام فإن المعالج ينفذ خطوة أو أمراً واحداً من البرنامج بعد إعطائه الأمر بذلك ثم ينتظر أمراً آخر لكي ينفذ الخطوة التالية وهكذا . إن المعالج بعد تنفيذ أي أمر يدخل في حالة الانتظار Waiting إلى أن يحيطه الأمر بتنفيذ الخطوة التالية . في أثناء حالة الانتظار تبقى آخر إشارة وضعت على المسارات كما هي ولا تتغير حتى أنه بعد الانتهاء من حالة الانتظار يستأنف المعالج عملية التنفيذ من نفس

المكان الذى توقف عنده . السؤال الآن كيف نستطيع إدخال المعالج فى حالة الانتظار هذه؟ إن ذلك يتم عن طريق الطرف رقم 35 من الشريحة 8085 وهو الطرف READY بمعنى جاهز أو مستعد . إن المعالج قبل تنفيذ أى أمر يقوم باختبار الطرف READY فإن كان هذا الطرف high يتم تنفيذ الأمر وإن كان الطرف READY فى حالة low فإن المعالج لن يتم تنفيذ الأمر وسيدخل فى حالة انتظار كما شرحنا . يجب هنا أن نفرق بين حالة الانتظار الناتجة من صفر يوضع على الطرف READY وحالة الـ HOLD التى رأيناها فى الجزء السابق . فى حالة HOLD ينفصل المعالج تماماً عن المسارات وتكون جميع المسارات فى وضع المقاومة العالية . أما فى حالة الانتظار التى نحن بصددها هنا فلا ينفصل المعالج عن المسارات ولكن تبقى المسارات حاملة لآخر إشارة تم وضعها على المسارات قبل أن يكون الخط READY صبراً . فى دائرة الميكروكمبيوتر التى سنبنيها نريد المعالج أن يكون فى حالة عمل مستمر ولذلك فسوف نضع خط READY فى حالة HIGH باستمرار أى غير فعال .

11-3-5 طرفى الحالة S1, S0

الإشارة الموجودة على هذين الخطين تبين حالة المعالج عند أى لحظة من اللحظات حيث أن المعالج لابد وأن يكون فى حالة من الأربع حالات الآتية :

1. حالة انتظار Waiting وهذه كما رأينا تكون عندما نضع صبراً أو low على الخط READY .

2. حالة كتابة سواء كانت كتابة فى ذاكرة أو بوابة إخراج .

3. حالة قراءة أمر من الذاكرة وتكون هذه فى أثناء دورة إحضار الأمر .

4. حالة قراءة معلومة سواء كانت المعلومة فى الذاكرة أو فى بوابة إدخال .

شكل (6-11) يبين الشفرة الموجودة على هذين الخطين فى مقابل كل حالة من الحالات الأربع . لاحظ أن الإشارة الموجودة على هذين الخطين تكون خارجة من المعالج ولذلك فإننا فى دائرة الميكروكمبيوتر التى سنبنيها سنترك هذين الخطين مفتوحين ولن نوصلهما بأى توصيلات خارجية .

حالة المعالج	S1	S0
انتظار Waiting	0	0
كتابة Write	0	1
قراءة Read	1	0
قرائة أمر Fetching	1	1

شكل (6-11) الحالات المختلفة للشريحة 8085

11-3-6 أطراف المقاطعة INTR, TRAP, RST7.5, RST6.5, RST5.5

يمكن استخدام خطوط المقاطعة لإيقاف المعالج من تنفيذ البرنامج الذي يقوم بتنفيذ الآن وجعله يذهب لتنفيذ برنامج آخر يسمى برنامج خدمة المقاطعة وبعد الانتهاء من تنفيذ برنامج خدمة المقاطعة يعود المعالج إلى البرنامج الأصلي حيث يستأنف تنفيذه من نفس المكان الذي حدثت عنده المقاطعة . لقد أفردنا فصلا كاملاً للمقاطعة لمن يريد تفاصيل وأمثلة عن هذا الموضوع ، وما يهمنا الآن هو ماذا سنفعل بهذه الخطوط الآن ؟ إذاً كنا نريد استخدام المقاطعة في البرنامج الذي سنستخدمه مع الميكروكمبيوتر الذي نتوى بناءه في هذا الفصل فعلينا أن نرجئ عملية بناء الدائرة لحين مراجعة الفصل الخاص بالمقاطعة ، أما إذا كنا لن نستخدم المقاطعة (كما هي الحال في المثال الذي نحن بصدده) فإن جميع هذه الخطوط يجب أن توضع في حالة عدم الفعالية وهذه الحالة تكون عند توصيل هذه الخطوط بالأرضي أي low كما هو مبين في الدائرة الكاملة للميكروكمبيوتر في آخر هذا الفصل .

11-3-7 الطرفان SOD, SID

الشريحة 8085 يمكن أن تستقبل منها أو ترسل إليها بيانات تتبعية Serial أي بت بعد بت وليس بنظام البایت كما عرفنا سابقاً . لن نتكلم عن عملية إدخال وإخراج البيانات تتبعياً في هذا الكتاب ولكن الذي يهمنا هنا هو أن نعرف أن البيانات التبعية تخرج من المعالج على الطرف SOD الذي يعني Serial Output Data بعد وضعها في مسجل التراكم وأما البيانات المراد إدخالها تتبعياً فإنها تدخل على الطرف SID الذي يعني Serial Input Data ومنه إلى مسجل التراكم . هذان الطرفان لن يكون لهما أي فعالية في دائرة الميكروكمبيوتر ذي الكارت الواحد ولذلك فإننا سنتركهما مفتوحان حيث أنهما لن يؤثرا على تشغيل المعالج بأى حال . لاحظ أن جميع الخطوط التي تحمل إشارة خرج من المعالج والتي لن يتم استخدامها يجب أن نتركها مفتوحة ولا يتم توصيلها بأى شيء (الأرضي أو ال Vcc) لأن ذلك يمكن أن يؤثر على عمل المعالج بما لا تحمد عقباه .

11-4-4 الأطراف الأخرى للمعالج Z80

11-4-1 أطراف المقاطعة RESET, BUSRQ, INT, NMI

جميع هذه الأطراف تعتبر مداخل للالمعالج يمكن مقاطعته من عليها وجميعها فعالة عند الصفر low active ، ولذلك فإنه طالما أن دائرة الميكروكمبيوتر ذي الكارت الواحد لن تستخدم المقاطعة أصلاً فإننا سنضع جميع هذه الخطوط في

حالة خمول بتوصيلها إلى Vcc أى high مع العلم أن فصل المقاطعة فى هذا الكتاب خاص بعمليات المقاطعة لمن يريد الإستزادة فى هذا الموضوع . الذى يهمنا معرفته هنا هو العنوان الذى يقفز إليه المعالج عند عمل RESET له أو عند إعادة القدرة إليه ؟ عند إعطاء نبضة على الطرف RESET يقفز المعالج Z80 إلى العنوان 0000H حيث يبدأ تنفيذ البرنامج الموجود فى هذا المكان ، لذلك يمكن توصيل هذا الطرف بدائرة كالمبينة فى شكل (4-11) والتى استخدمناها لعمل RESET للبروسيسور 8085 . الخطان BUSACK و BUSRQ يكافيان الخطان HOLD و HLDA فى حالة المعالج 8085 حيث يمكن لأى جهاز خارجي بجعل الطرف BUSRQ فعالا low أن يطلب من المعالج التخلى عن المسارات ووضعها فى الحالة المنطقية الثالثة حتى يت Sensors للجهاز الخارجى إستعمالها فى التعامل المباشر مع الذاكرة مثلا . BUSRQ تعنى طلب المسارات Bus Requist عندما يتخلى المعالج عن المسارات يقوم بوضع صفر low على الخط BUSACK والذى منه يعرف الجهاز الخارجى أن المعالج قد تخلى فعلا عن المسارات وتركها وكلمة BUSACK تعنى اعترافا بطلب التخلى عن المسارات Bus Acknowledge . فى حالة استخدام المعالج Z80 فى دائرة الميكروكومبيوتر المقترحة يجب أن يوضع الخط BUSRQ فى حالة خمول أو عدم فعالية بتوصيله على أى Vcc . أما الخط BUSACK فطالما أنه خط خرج فيجب أن يترك مفتوحا open ولا يوصل بشيء .

2-4-2 الطرف RFSH

يستخدم هذا الخط فى عملية تجديد محتويات الذاكرة الديناميكية dynamic memory كل فترة زمنية محددة ، وطالما أن هذا الخط يعتبر خط خرج أى يحمل إشارات خارجة من المعالج فإننا سنتركه مفتوحا طالما أنشأنا لن نستخدم هذا النوع من الذاكرة فى الدائرة المقترحة ولا ننوى الدخول فى تفاصيل عملية المواجهة مع الذاكرة الديناميكية هنا .

2-4-3 الطرف HALT

الخط HALT خط خرج ويكون فعالا عندما يكون المعالج فى حالة HALT أى توقف عن العمل بعد تنفيذ الأمر HALT ولا يخرج المعالج من هذه الحالة إلا عن طريق مقاطعة وطالما أنه خط خرج فسنتركه مفتوحا open .

2-4-4 الطرف WAIT

يستخدم هذا الخط فى إدخال المعالج فى حلقة إنتظار حيث يتوقف فيها المعالج عن تنفيذ أى أمر ولا ينفصل عن المسارات ويمكن استخدام هذا الطرف فى تنفيذ

البرامج بنظام الخطوة خطوة مثل الطرف READY في المعالج 8085 وأيضا عندما تكون الأجهزة المحيطة غير مستعدة للتعامل مع المعالج بسبب فارق السرعة مثلا . هذا الخط يكون فعالا عندما يكون صفرأً أي أنه low ولذلك فإننا يجب أن نوصله على Vcc في دائرة الميكروكمبيوتر المقترحة لنجعله غير فعال .

5-4-5 الطرف M1

هذا الخط هو خط خرج يبين حالة المعالج حيث يكون فعالا عندما يكون المعالج في حالة إحضار الأوامر من الذاكرة وهذا الخط لن نستخدمه في دائرتنا ولذلك سنتركه مفتوحا open .

5-4-6 الطرف CLK

يستخدم هذا الخط لإدخال نبضات التزامن الخاصة بالمعالج وهي نبضات TTL يتراوح ترددتها بين 2.5MH أو 4MH أو 6MH وذلك على حسب نوع المعالج المستخدم .

بذلك تكون قد انتهينا من التعرف على جميع الأطراف الأخرى للبروسيسور Z80 وعرفنا فكرة موجزة عنها ونستطيع أن نلخص القول في أن جميع خطوط الخرج الغير مستخدمة يجب أن تترك مفتوحة open وجميع خطوط الدخل الغير مستخدمة توسيع في وضع خمول أي عدم فعالية .

5-5 إشارات المرور

الآن بعد أن عرفنا وظيفة كل طرف من أطراف المعالج وماذا سنفعل بكل طرف من الأطراف الغير مستخدمه ، ماذا عن دائرة المعالج التي ستتحكم في إشارة المرور كما ذكرنا في المثال التوضيحي ؟ قبل أن ندخل في تفاصيل بناء أو حل هذه المسألة سنتفق بعض الوقت في التفكير في خطة الحل . إن حل هذه المسألة كباقي المسائل التي تستخدم المعالج يتكون من جزء بناء الدائرة hardware وجاء برمجة software . بالنسبة لجزء البناء وكما أشرنا في بداية الفصل فإننا سنحتاج لما يلى :

1. المعالج وقد تم فصل جميع مساراته ، مسار البيانات D0 إلى D7 ومسار العناوين A0 إلى A15 ومسار التحكم بالإضافة إلى خط إعادة الوضع . سنستخدم كما ذكرنا المعالج 8085 كمثال ومن يريد استخدام أي معالج آخر فذلك أصبح سهلا جدا بعد قراءة هذا الفصل . ولقد رأينا في الفصل الثامن كيفية فصل

- المسارات الثلاثة ويمكن مراجعة ذلك وبالذات للمعالج 8085 الذي سنستعمله في هذا المثال . شكل (11-7) يبين المعالج والشريحة المستخدمه في عملية الفصل .
2. إشارة المرور الرباعية كما رأينا بها 12 لمبة (ثلاثة في كل ركن من أركان القطاع ، أحمر وأخضر وأصفر) ومطلوب إدارة هذه اللمبات بتتابع معين وأزمنة محسوبة كما سنرى بعد قليل . لذلك سنحتاج إلى بوابتي إخراج بمجموع 16 بت سنستخدم منهم 12 لإدارة ال 12 لمبة ويتبقي 4 برات من ال 16 لـن تستخدـم وستترك للاستخدام المستقبلي . كما ذكرنا سابقا في المثال التوضيحي فإن هناك مفتاح سيقوم المعالج بقراءته دائمـا وإذا كان هذا المفتاح واحدـا (ON) فإن الإشارة تعمل في الحالة العادية ، وأما إذا كان المفتاح صفرـا (OFF) فإن الإشارة ستعمل في الحالة الترددية للون الأصفر flashing . لذلك فإنـنا سنحتاج إلى بوابة إدخـال سنستعمل منها بت واحدة لقراءة حالة هذا المفتاح والباقي لن يستخدم وسيترك لأى استخدامات مستقبلـية . خلاصة القول لأنـنا سنحتاج إلى بوابـتـي إخراج وبـوـابـة إدخـال . بعد أن قررـنا أنـنا سنحتاج إلى بوابـتـي إخراج وبـوـابـة إدخـال يجب أن نقرر أيضا بأى طريقة من الطرق التـى درسـناها في الفـصل العـاشر سـنبـنى هـذـه الـبـوابـات . لقد قررـنا نـحن أنـ نـسـتـخـدـم الـبـوابـات الـقـابـلـة لـلـبـرـمـجـة أـى الشـرـيـحة 8255A وذلك لـسـهـولة بـرـمـجـتها وـبـنـائـها وـكـوـنـها شـرـيـحة وـاحـدة تـحـتـوى الـثـلـاثـة بـوـابـاتـ الـتـى نـحتاج إـلـيـها . شـكـل (11-8) يـبـيـن طـرـيقـة توـصـيل هـذـه شـرـيـحة عـلـى المسـارـاتـ الـثـلـاثـةـ الـقادـمةـ مـنـ الـمعـالـجـ .
3. يـبـيـن شـكـل (11-8) أيضـا كـيفـية توـصـيل شـرـيـحة الـذاـكـرـة EPROM وهـى شـرـيـحة 2716 الـتـى سـنـكـتـب عـلـيـها بـرـنـامـج التـحـكـم فـى كـلـ الـعـمـلـيـة . لـقد درـسـنا فـى الفـصل التـاسـع طـرـيقـة توـصـيل الـذاـكـرـة عـلـى الـمعـالـجـ وـالـعـمـلـيـة هـذـه أـبـسـطـ بـكـثـيرـ مـمـا شـرـحـنا فـى الفـصل التـاسـع ، فإـنـه طـالـمـا أنـ الـمعـالـجـ سـيـكـون مـتـصـلـا بـشـرـيـحة ذـاـكـرـة وـاحـدةـ فـىـ اـعـلـىـ الـعـنـوـنـةـ وـالـتـشـفـيرـ مـنـ الـمـمـكـنـ أنـ تـكـونـ بـسـيـطـةـ جـداـ إـذـاـ وـصـلـنـا خطـ تـشـيـطـ الشـرـيـحة 2716 وهوـ الخطـ CS عـلـىـ خـطـ التـحـكـم MEMR وـخطـ تـشـيـطـ خـرـجـ الشـرـيـحة OE بالـأـرـضـىـ مـباـشـرـةـ . فـىـ هـذـهـ الـحـالـةـ فـىـ الشـرـيـحةـ سـتـعـملـ وـتـخـرـجـ خـرـجـهـاـ مـعـ أـىـ أمرـ قـرـاءـةـ مـنـ الـذـاـكـرـةـ وـلـاـ حـرـجـ فـىـ ذـلـكـ حـيـثـ أـنـهـاـ هـىـ شـرـيـحةـ الـذـاـكـرـةـ الـوـحـيدـةـ الـمـوـصـلـةـ مـعـ الـمـعـالـجـ . لـاحـظـ أـنـهـ طـالـمـاـ أـنـ الـEPROMـ سـتـعـملـ مـعـ أـىـ عنـوانـ يـخـرـجـهـ الـمـعـالـجـ فإـنـهـ يـجـبـ أـنـ نـتـوـقـ أـنـهـ سـتـعـملـ بـمـجـرـدـ توـصـيلـ الـقـدرـةـ لـلـدـائـرـةـ لـأـنـ الـمـعـالـجـ 8085ـ كـمـاـ رـأـيـناـ سـابـقاـ عـنـدـ توـصـيلـ الـقـدرـةـ إـلـيـهـ أوـ إـعادـةـ وـضـعـهـ RESETـ فإـنـهـ يـبـدـأـ التـفـيـذـ مـنـ الـعـنـوانـ 0000Hـ حـيـثـ سـيـشـغـلـ الـEPROMـ .
4. يمكن توـصـيلـ كـلـ بـتـ منـ بـتـاتـ الـخـرـجـ الـخـارـجـةـ مـنـ الـشـرـيـحةـ 8255ـ عـلـىـ قـاعـدـةـ تـرـانـزـسـتـورـ لـيـعـمـلـ كـفـاـصـلـ bufferـ لإـدـارـةـ هـذـهـ الـدـايـوـدـاتـ حـتـىـ تعـطـىـ كـمـيـةـ إـضـاءـةـ أـحـسـنـ . شـكـل (11-8) يـبـيـنـ ذـلـكـ أـيـضاـ .

5. يمكن توصيل خرج البوابات على لمبات 220 فولت للحصول على إنارة قوية ودائمة واقعية عن طريق استخدام عوازل صوتية opto-couplers . بالنسبة لجزء البرمجة software فإننا سنفكر فيه بالطريقة التالية :

1. يحتوى شكل (11-9) على جدول به جميع الحالات الممكنة لجميع اللعبات وزمن كل حالة والشفرة أو الرقم المطلوب إخراجه على بوابات الإخراج للحصول على هذه الحالة . من شكل (11-9) نلاحظ مثلا أنه في الحالة الأولى يكون الأخضر الأول مضيئا والأحمر في جميع الاتجاهات الأخرى مضيئا أيضا والأصفر في جميع الاتجاهات مطفأً وسوف يستمر هذا الوضع لمدة 50 ثانية . لاحظ أن المضيء في الجدول يساوى واحدا والمطفأ يساوى صفراء ، لذلك فإن هذا الوضع الأول سيكافئ الشفرة 4C على بوابة الإخراج الأولى والشفرة 02 على بوابة الإخراج الثانية . بعد ذلك يتضيء الأصفر الأول مع الأخضر الأول مع نفس الوضع السابق لجميع اللعبات الأخرى وذلك لـ 10 ثوان وذلك كزمن تحذير بالوقوف بعد الضوء الأخضر . بعد ذلك ستضيء جميع اللعبات الحمراء في جميع الاتجاهات لمدة 5 ثوان كزمن أمان خوفا من السيارات المسرعة التي لا تستطيع التوقف في خلال الضوء الأصفر . بعد ذلك تكرر هذه الأذمنة لجميع الاتجاهات الأخرى .

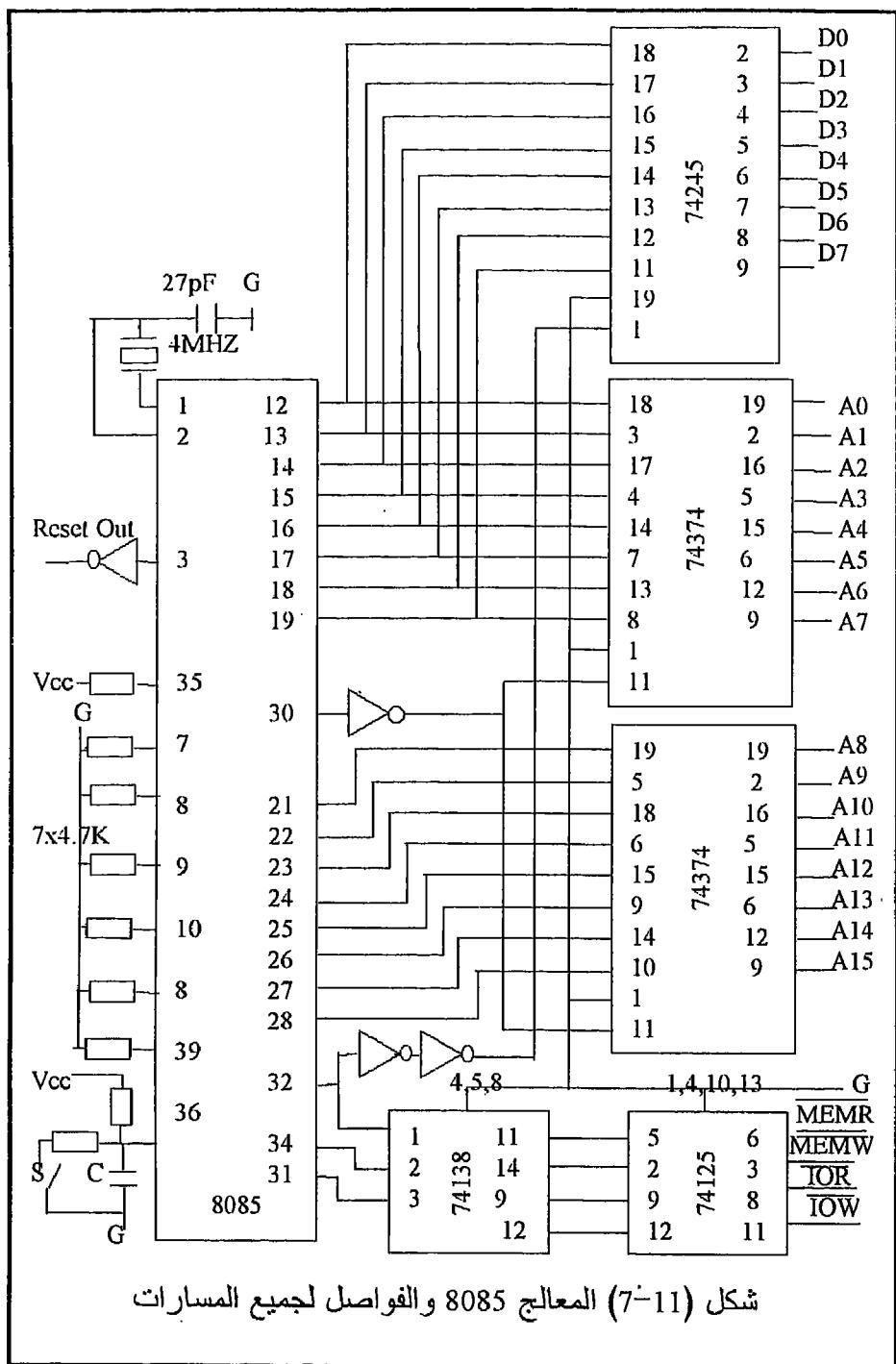
لاحظ أن هذه الأذمنة يمكن التحكم فيها بالزيادة والتقصان على حسب الرغبة ، ثم إن تتبع الضوء الأخضر في الاتجاهات المختلفة بمعنى أن أي اتجاه سيسمح له بالمرور وأى اتجاه سيسمح له بعده ، هذه أيضا يمكن التحكم فيها . إن فكرة البرنامج التى نقترحها هنا (بالطبع فإن كل قارئ ستكون لديه فكرة مختلفة وربما أفضل) تعتمد على ما يلى :

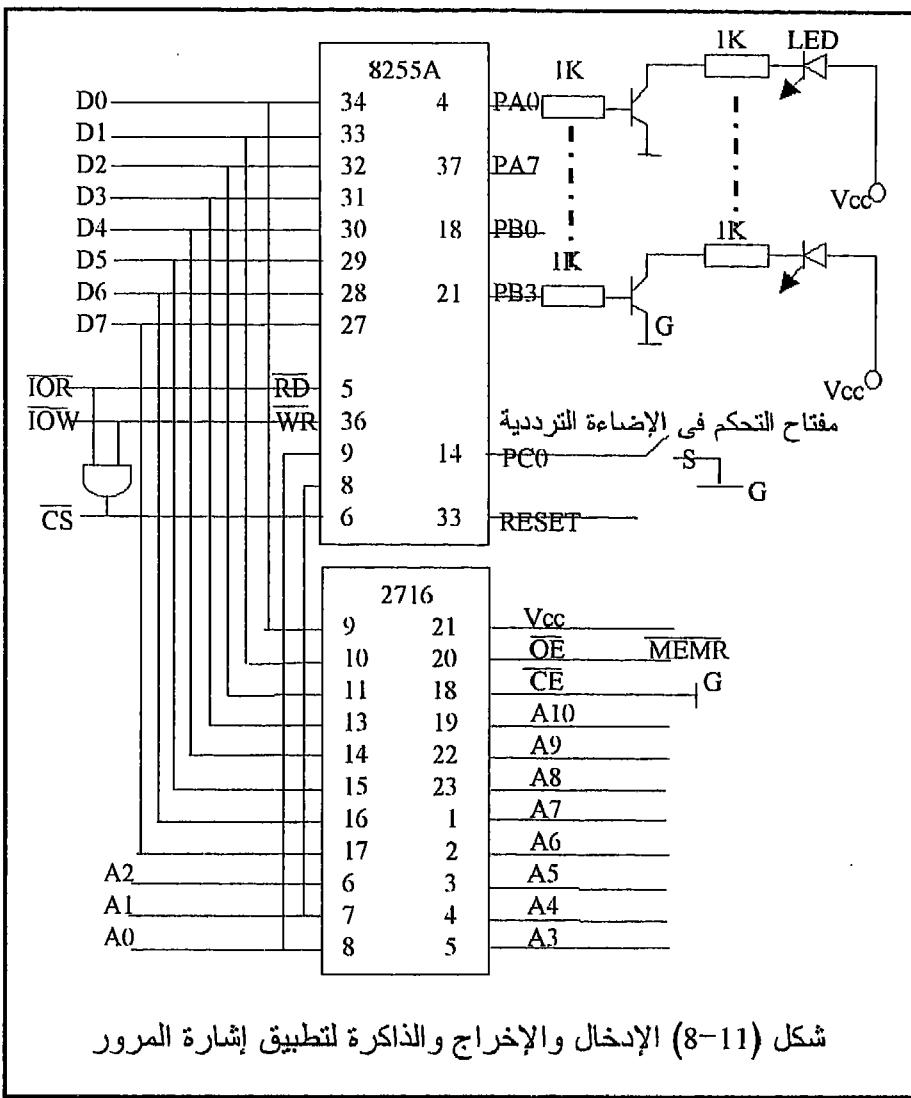
1. سنخزن في مكان ما فى الذاكرة الشفرات المطلوب إخراجها على بوابات الإخراج لكل حالة وكذلك زمن كل حالة بالتتابع من الجدول المبين فى شكل (11-9) ولتكن ذلك كما يلى :

4 ستخرج على البوابة الأولى	4C , E100 4C
, 02 ستخرج على البوابة الثانية	E101 02
, 50 زمن تأخير للحالة الأولى	E102 50
, 4E تخرج على البوابة الأولى	E103 4E
, 02 تخرج على البوابة الثانية	E104 02
, 10 هي زمن تأخير للحالة الثانية	E105 10

وهكذا سيحتوى هذا الجدول على 12 بait .

2. سيكون البرنامج عبارة عن قراءة للشفرتين المقابلتين لكل حالة وإخراجهما على بوابتي الإخراج رقم 00 ورقم 01 ثم الدخول فى زمن التأخير المقابل لهذه الحالة .





3. قبل الدخول فى أى حالة جديدة لابد أن يقرأ المعالج بوابة الإدخال رقم 02 ليعرف إذا كان مفتاح التحكم المقابل للبت رقم صفر يساوى واحد أم صفر ، فإذا كانت هذه البت تساوى واحد فلن البرنامج يسير فى سيره الطبيعي كما فى الجدول المبين فى شكل (11-9) ، وإذا كانت هذه البت تساوى صفرًا سيقفز إلى برنامج آخر ينفذ عملية التردد على اللون الأصفر .

شكل (11-10) يبين خريطة التدفق ، والبرنامج مكتوب بلغة الأسمبلى للبروسيسور 8085 سيكون كما يلى :

(يلاحظ من هذا البرنامج خلوه من البرامج

الفرعية subroutines بالرغم من أفضلية استخدامها هنا في مثل هذه التطبيقات ولكننا تجنبنا ذلك لسببين أولهما أنها مازلنا لم نشرح البرامج الفرعية حتى الآن في هذا الكتاب وثانيهما أن الدائرة التي بنيناها ليس بها ذاكرة RAM تستخد كمكذبة stack في حالة استخدام البرامج الفرعية كما سنرى).

الزمن بالثانية	البوابة 01 00	G4Y4R4	G3Y3R3	G2Y2R2	G1Y1R1	رقم الحالة
50	02 4C	0 0 1	0 0 1	0 0 1	1 0 0	1
10	02 4E	0 0 1	0 0 1	0 0 1	1 1 0	2
5	02 49	0 0 1	0 0 1	0 0 1	0 0 1	3
50	02 61	0 0 1	0 0 1	1 0 0	0 0 1	4
10	02 71	0 0 1	0 0 1	1 1 0	0 0 1	5
5	02 49	0 0 1	0 0 1	0 0 1	0 0 1	6
50	03 09	0 0 1	1 0 0	0 0 1	0 0 1	7
10	03 89	0 0 1	1 1 0	0 0 1	0 0 1	8
5	02 49	0 0 1	0 0 1	0 0 1	0 0 1	9
50	08 49	1 0 0	0 0 1	0 0 1	0 0 1	10
10	0C 49	1 1 0	0 0 1	0 0 1	0 0 1	11
5	02 49	0 0 1	0 0 1	0 0 1	0 0 1	12
5	02 49	من هنا يبدأ تكرار هذه الحالات				
						1

شكل (11-9) جدول الحالات المختلفة لإشارة المرور

البوابة A أخراج و B أخراج و C إدخال والجميع في الحالة صفر

MVI A,89H ;

إرسال إلى مسجل التحكم في 8255A

OUT 03H ;

مسجل C عداد يحتوى 12 بايت

START: MVI C,0CH ;

مؤشر الذاكرة ابتداء من E100 HL

LXI H,E100 ;

قراءة البوابة C

READ: IN 02H ;

حجب لجميع البيانات ما عدا الأولى

ANI 01H ;

قفز عندما PC0=0 لترنذ اللون الأصفر

JZ FLASH ;

إحضار الشفرة الأولى من الذاكرة

MOV A,M ;

إخراج هذه الشفرة على البوابة A

OUT 00 ;

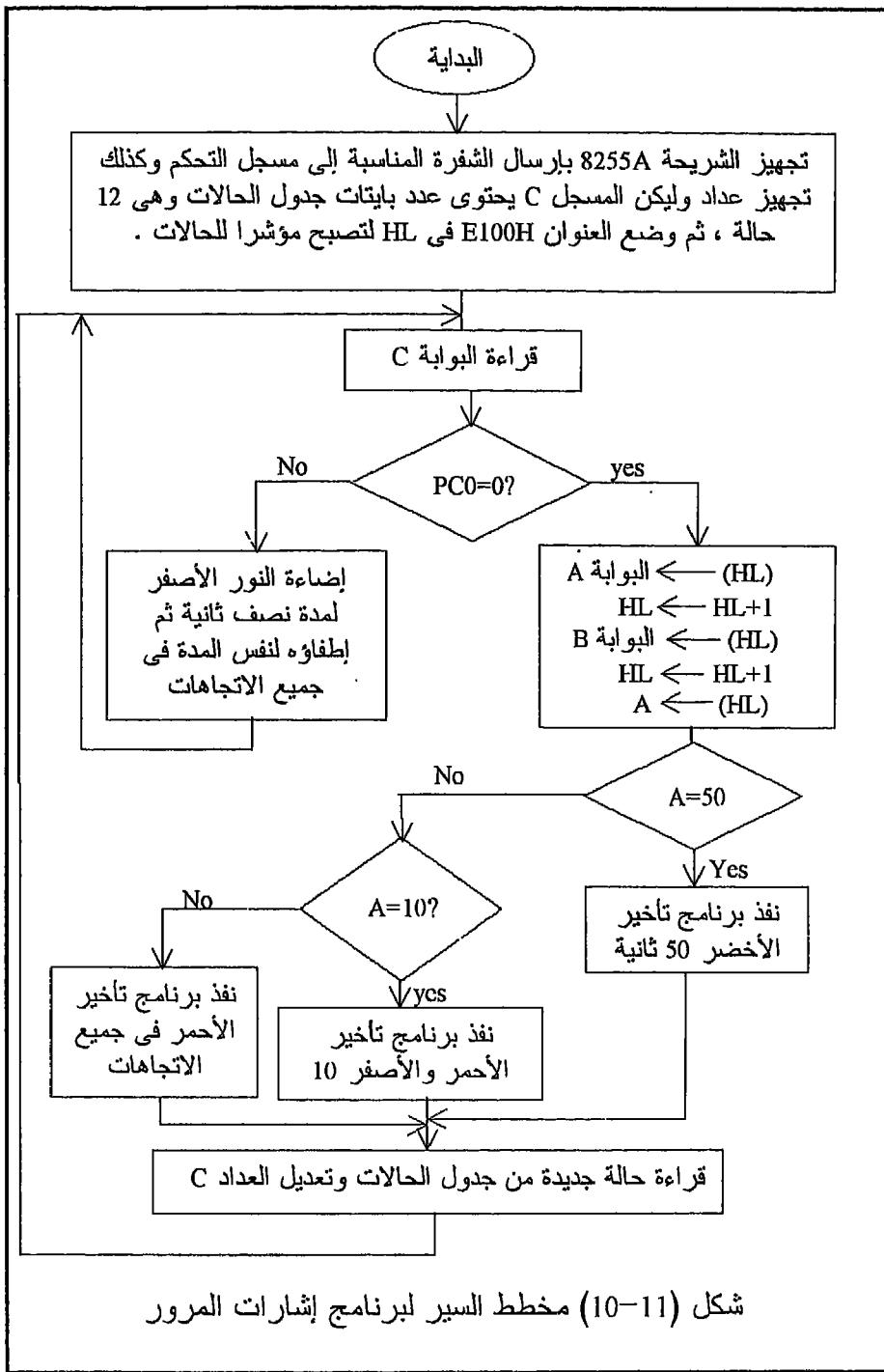
إحضار الشفرة الثانية من الذاكرة

INX H ;

إخراج هذه الشفرة على البوابة B

MOV A,M ;

INX H ;



شكل (11-10) مخطط السيর لبرنامج إشارات المرور

MOV A,M ;	إحضار زمن تأخير
CPI 50H ;	هل هذا الزمن = 50
JNZ TEN ;	قفز إذا لم يكن الزمن = 50
MVI B,6CH ;	بداية برنامج تأخير 50 ثانية
LOOP1: MVI D,FFH ;	
LOOP2: MVI A,FF ;	
LOOP3: DCR A ;	
JNZ LOOP3 ;	
DCR D ;	
JNZ LOOP2 ;	
DCR B ;	
JNZ LOOP1 ;	
JMP CONTIN ;	
TEN: CPI 10H ;	هذا التأخير لا يساوى 50 فهل = 10
JNZ FIVE ;	قفز إذا لم يكن التأخير = 10
MVI B,16H ;	بداية برنامج تأخير 10 ثوان
LOOP4: MVI D,FFH ;	
LOOP5: MVI A,FFH ;	
LOOP6: DCR A ;	
JNZ LOOP6 ;	
DCR D ;	
JNZ LOOP5 ;	
DCR B ;	
JNZ LOOP4 ;	
JMP CONTIN ;	
FIVE: MVI B,0BH ;	هذا التأخير لا يساوى 10 ولكن يساوى 5 ثوان
LOOP7: MVI D,FFH ;	بداية برنامج تأخير 5 ثوان
LOOP8: MVI A,FFH ;	
LOOP9: DCR A ;	
JNZ LOOP9 ;	
DCR D ;	
JNZ LOOP8 ;	
DCR B ;	
JNZ LOOP7 ;	
CONTIN: DCR C	
JZ START ;	
INX H ;	
JMP READ ;	
FLASH: MVI A,92H ;	بداية برنامج تردد الضوء الأصفر

OUT 00H ;	الرقمان 92H و 04H يضيئن الأصفر
MVI A,04H ;	في جميع الاتجاهات
OUT 02H ;	
MVI B,D8H ;	بداية تأخير نصف ثانية
LOOPA: MVI D,FFH ;	
LOOPB: DCR D ;	
NOP ;	
JNZ LOOPB ;	
DCR B ;	
JNZ LOOPA ;	
MVI A,00H ;	إطفاء النور الأصفر في جميع الاتجاهات
OUT 00H ;	
OUT 01H ;	
MVI B,D8H ;	بداية تأخير نصف ثانية
LOOCP: MVI D,FFH ;	
LOOPD: DCR D ;	
NOP ;	
JNZ LOOPD ;	
DCR B ;	
JNZ LOOCP ;	
JMP READ ;	قفز لقراءة بوابة الإدخال PC

- سنرى في فصل البرامج الفرعية كيفية الحصول على أزمنة التأخير باستخدام البرامج الفرعية مما سيؤدى إلى اختصار خطوات مثل هذا البرنامج بدرجة كبيرة ، ويمكن إعادة كتابة البرنامج بهذا الوضع كتمرين على البرامج الفرعية .

11-6 تمارين

1. أعد تصميم كارت الميكروكمبيوتر ، وكذلك البرنامج ، الخاصين بالتحكم فى إشارة المرور ولكن مستخدما هذه المرة المعالج Z80 بدلا من المعالج 8085 كما كان فى هذا الفصل .

الفصل الثاني عشر

البرامج الفرعية

Subroutines

1-12 مقدمة

نستطيع القول بأنه عامة عندما تواجهك كمبرمجة مشكلة كبيرة ومطلوب منك برمجتها فإن أسهل الطرق لذلك هي أن تقوم بتجزئه أو تكسير هذه المشكلة الكبيرة إلى مشاكل أو مسائل أصغر ثم تقوم ببرمجة هذه المسائل الصغيرة كل على حدة ثم يكون هناك برنامج أساسى يقوم بتحميص أو تنفيذ هذه الأجزاء الصغيرة بالتتابع الذى يحل المسألة أو المشكلة الأساسية . أحد طرق التجزئه هذه هي البرامج الفرعية subroutines . إن استخدام البرامج الفرعية من مميزاتها تسهيل عملية البرمجة واختصار كمية الذاكرة المستخدمة لكتابه شفرات البرنامج كما سنرى فى هذا الفصل .

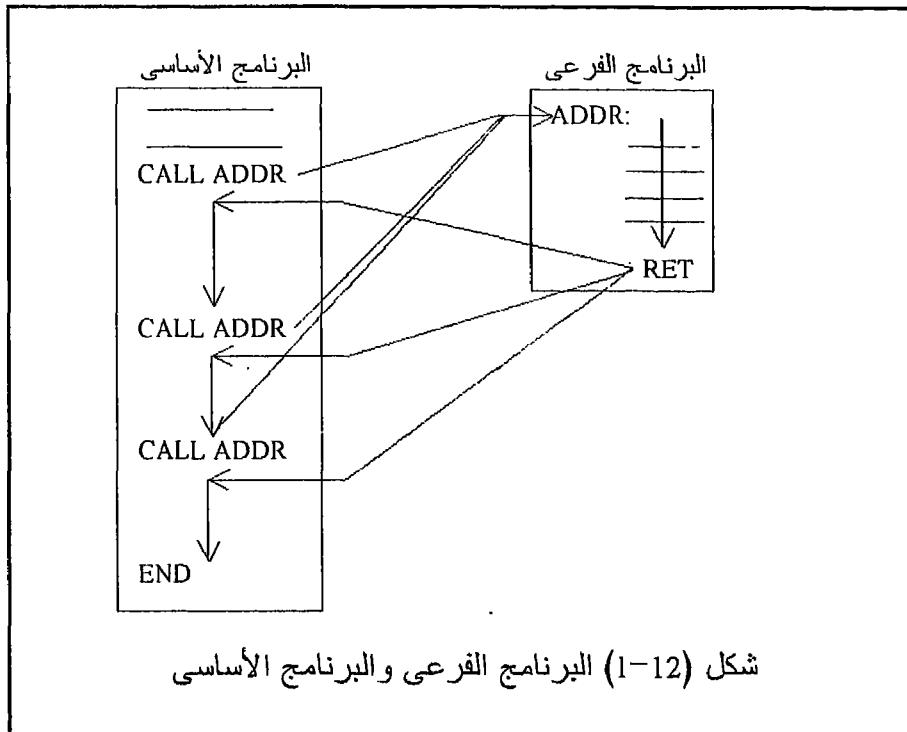
2-12 ما هو البرنامج الفرعى ؟

شكل (1-12) يبين رسمياً توضيحاً لعلاقة البرنامج الفرعى بالبرنامج الأساسي . نلاحظ من هذا الشكل أن البرنامج الفرعى عبارة عن جزء من برنامج ، أو برنامج صغير ، يتم النداء عليه للتنفيذ من البرنامج الأساسي فينفذ ، وبعد الانتهاء من تنفيذه تتم العودة إلى البرنامج الأساسي وعند نفس المكان الذى تم الخروج منه للبرنامج الفرعى . أى أن طريقة تنفيذ البرنامج الفرعى تشابه وإلى حد كبير طريقة تنفيذ أوامر القفز ، الاختلاف فقط هو في عملية العودة إلى نفس المكان الذى تم القفز منه في البرنامج الأساسي بعد الانتهاء من تنفيذ البرنامج الفرعى ، ويرجع ذلك إلى بعض الخطوات أو الاحتياطات التى يعملاها المعالج قبل القفز إلى البرنامج الفرعى .

من شكل (1-12) نستطيع أن نتبين الفائدة العظيمة من استخدام البرنامج الفرعية وهى توفير الذاكرة المستخدمة لكتابه البرنامج . في الكثير من التطبيقات يكون هناك جزء من البرنامج تكون مسيطرة لكتابته أكثر من مرة وكمثال على ذلك جزء البرنامج الذى يعمل زمان التأخير في برنامج إشارات المرور في الفصل السابق . إن مثل هذا الجزء باستخدام البرنامج الفرعية يمكن كتابته مرة واحدة فقط وفي كل مرة تكون هناك الحاجة إلى تنفيذه يتم النداء عليه بالأمر CALL فينفذ وبعد الانتهاء من تنفيذه ترجع عملية التنفيذ إلى حيث خرجت من البرنامج الأساسي .

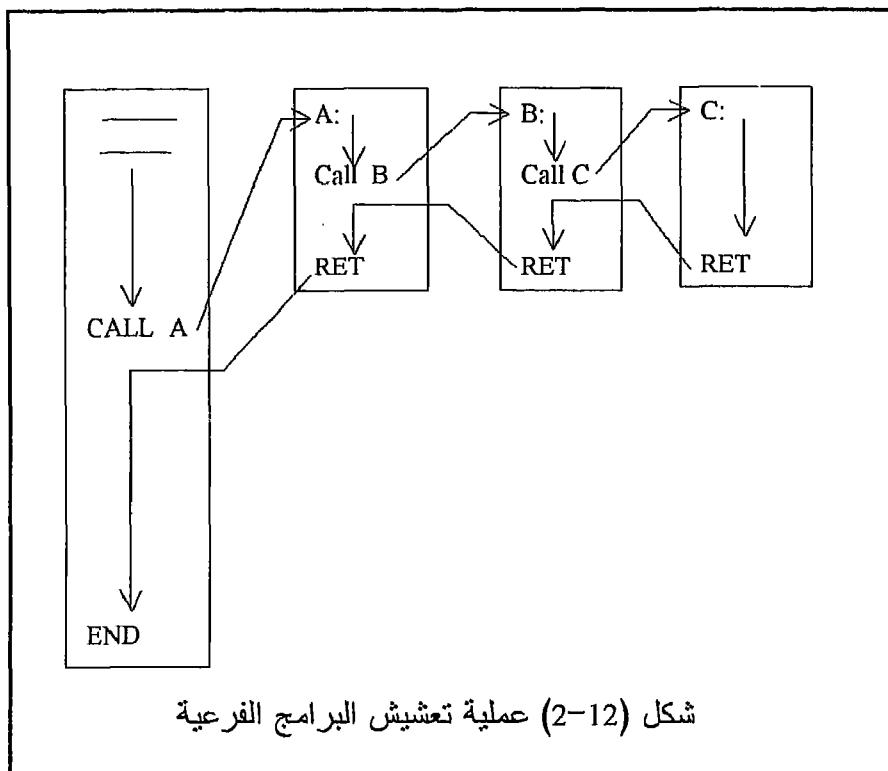
شكل (2-12) يبين خاصية أخرى في البرامج الفرعية وهي أن أي برنامج فرعى يمكنه النداء على برنامج فرعى آخر ، فمثلاً البرنامج الأساسي ينادى البرنامج الفرعى (أ) والبرنامج الفرعى (أ) ينادى البرنامج الفرعى (ب) والبرنامج الفرعى (ب) ينادى البرنامج الفرعى (ج) وهكذا لأى عدد من .

التدخلات . هذه العملية تسمى عملية تعشيش nesting للبرامج الفرعية . بعد الانتهاء من تنفيذ آخر برنامج فرعى فى السلسلة وليكن البرنامج الفرعى (ج) فإن المعالج يرجع إلى البرنامج الفرعى (ب) من حيث تم النداء على البرنامج الفرعى (ج) وتنتمى تكملة البرنامج الفرعى (ب) حيث يرجع المعالج إلى البرنامج الفرعى (أ) من حيث تم النداء على البرنامج الفرعى (ب) ، بعد الانتهاء من تنفيذ البرنامج الفرعى (أ) تتم العودة إلى البرنامج الأساسى من حيث تم النداء على البرنامج الفرعى (أ) .



يجب أن نحذر خطأً أو فخاً يمكن أن نقع فيه وهو أن ينادى واحد من البرامج الفرعية اللاحقة أحد البرامج الفرعية السابقة كأن ينادى مثلاً البرنامج (ج) البرنامج (ب) أو (أ) . في هذه الحالة سيدور المعالج في حلقة لانهائية لا مخرج منها ولن يرجع المعالج أبداً إلى البرنامج الأساسى الذى خرج منه حيث سيظل البرنامج (ج) ينادى على (ب) والبرنامج (ب) ينادى على (ج) إلى ما لا نهاية . هناك الكثير من أوامر النداء على والعودة من البرامج الفرعية . فمنها ما هو غير مشروط مثل الأمر CALL addr للشريحتين 8085 و Z80 بحيث ينتقل التنفيذ إلى العنوان addr دون أي شرط مثله في ذلك مثل الأمر JMP . وفي

المقابل هناك أمر العودة RET للمعالج 8085 و Z80 الذى يكون آخر أمر فى البرنامج الفرعى والذى عند تنفيذه تتم العودة دون أى شرط إلى المكان الذى تم النداء منه . هناك أيضا النداء المشروط على البرامج الفرعية والعودة المشروطة من البرامج الفرعية مثل الأمر CZ addr للمعالج 8085 والذى يعنى نداء على برنامج فرعى مشروط بعلم الصفر يساوى واحدا . كما أن هناك أيضا العودة المشروطة بعلم الصفر يساوى واحدا وهو الأمر RZ للمعالج 8085 . راجع الفصل الخاص ببرمجة المعالج الذى تستخدمه للتعرف على جميع أوامر النداء والعودة من البرامج الفرعية .



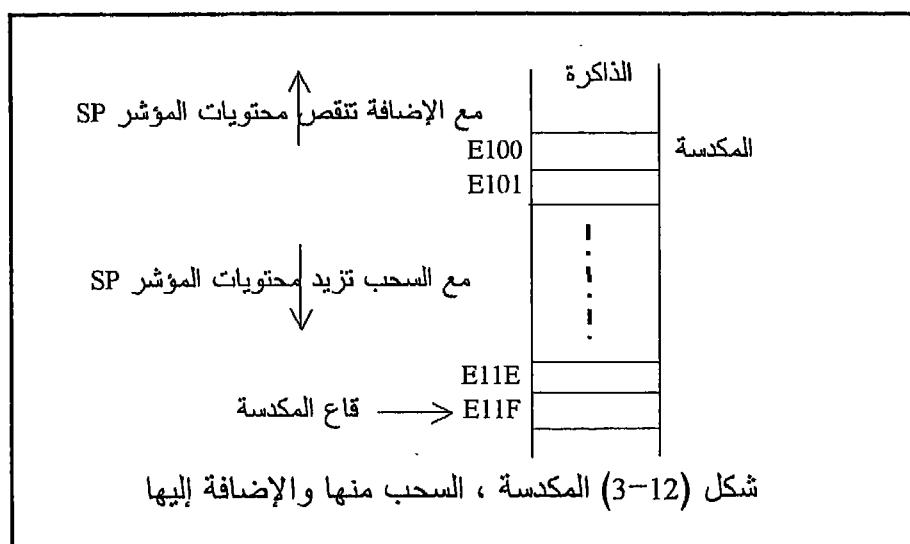
شكل (12-2) عملية تعشيش البرامج الفرعية

12-3 كيف يعود المعالج إلى نفس المكان الذي خرج منه ؟

إن السر يكمن في المكدسة stack ومؤشر المكدسة stack pointer . المكدسة هي جزء مقتطع من الذاكرة RAM الملحة على المعالج لخدمة أغراض النداء والعودة من البرامج الفرعية وأيضا لخدمة أغراض المقاطعة وأغراض أخرى كما سنرى

في فصل آخر . إن أقرب تشبيه للمكدة هو الجوال (الشوال) الذى نضيف إليها من فوهته وعندما نأخذ منه فإننا نأخذ من فوهته أيضا ، أي أن آخر ما وضعنا فى الشوال (المكدة) يكون أول ما نأخذ منها أو Last In First Out وتختصر LIFO . تشبيه آخر للمكدة هو رص الأطباق ، فأنك حينما ترص الأطباق رأسيا تضيف إلى قمة المكدة وعندما تريد أخذ طبق فإنك تأخذ آخر طبق وضعه على القمة .

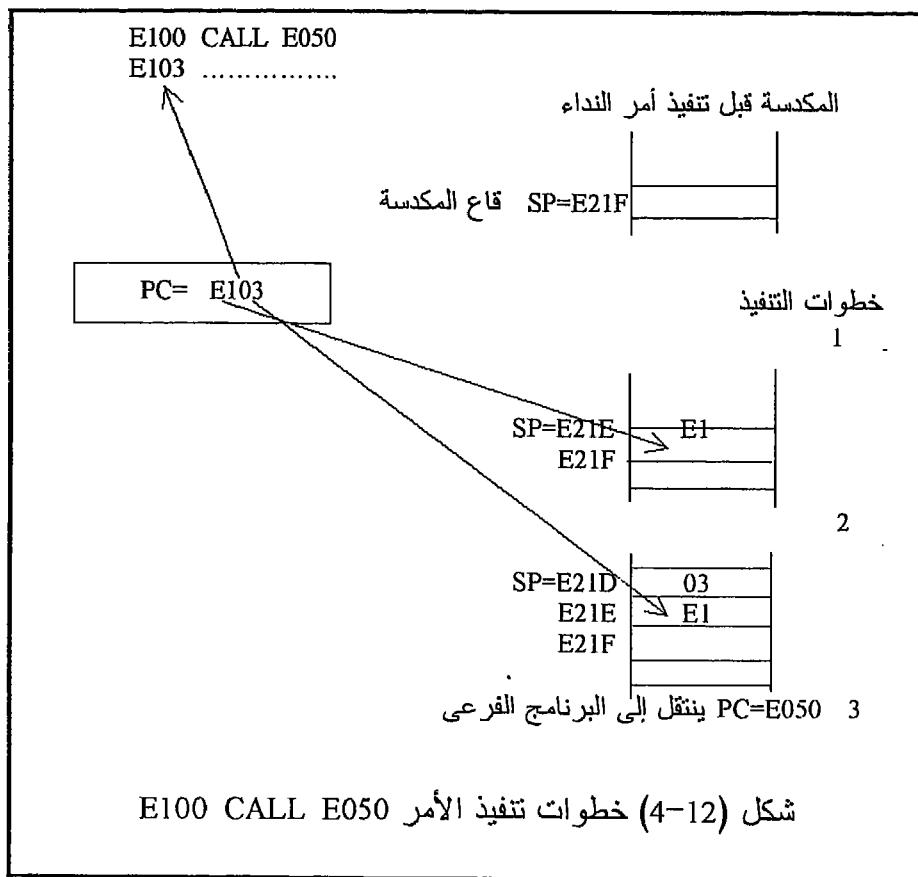
وأما مؤشر المكدة SP فإنه مسجل مكون من 16 بت محتوياته هى عنوان قمة أو آخر مكان تم التخزين فيه فى المكدة . عندما تكون المكدة فارغة فإن مؤشر المكدة يشير إلى قاعها وتكون محتويات ال SP هى عنوان آخر مكان فى المكدة . عند الإضافة إلى المكدة فإن المؤشر ينقص محتوياته وعند السحب من المكدة فإن المؤشر تزيد محتوياته بحيث أن الزيادة أو النقص تكون دائما بمقادير 2 لكل عملية سحب أو إضافة كما فى شكل (12-3) .



إن الإضافة والسحب من المكدة تكون دائما على أزواج المسجلات ولا يمكنها بأى حال أن تضيف أو تسحب مسجلا واحدا فقط أو بait واحده فقط من أو إلى المكدة . لذلك فإن مؤشر المكدة حينما يزيد أو ينقص فإنه يزيد أو ينقص بمقادير اثنين ، أي اثنين bait ، ولا يمكن أن يزيد أو ينقص بمقادير واحد على الإطلاق . شكل (12-4) يبين الخطوات التى يقوم بها المعالج عند تنفيذ أمر الداء على أي برنامج فرعى وهى كالتالى :

1. مؤشر المكدة SP ينقص decrement بمقدار واحد ، ويخزن البait ذات

- القيمة العظمى من عدد البرنامج PC في هذا العنوان .
2. مؤشر المكدة SP ينقص بقدر واحد آخر وتخزن البايت ذات القيمة الصغرى من عدد البرنامج في العنوان الجديد . بذلك يكون قد تم الحفاظ على محتويات عدد البرنامج وهي عنوان الأمر الذى عليه الدور فى التنفيذ بعد الأمر CALL في المكدة .
3. يحمل عدد البرنامج بعنوان أول بايت في البرنامج الفرعى وهو العنوان الموجود في أمر النداء CALL addr . بذلك ينتقل التنفيذ إلى البرنامج الفرعى . لاحظ من شكل (12-4) أن قاع المكدة وهي العنوان E21F تكون فارغة دائماً وذلك لأن عملية إنقاص مؤشر المكدة تكون دائماً قبل التخزين فيها . في نهاية تنفيذ البرنامج الفرعى يكون آخر أمر ينفذ هو الأمر RET وعند تنفيذ هذا الأمر تتم العمليات العكسية للعمليات الموضحة في شكل (12-4) وهي كالتالى :



1. محتويات قمة المكدة في العنوان E21D وهي الرقم 03 تدفع إلى البايت

ذات القيمة الصغرى من عداد البرنامج وتزداد قيمة محتويات مؤشر المكدة بمقدار واحد فيصبح E21E .

2. محتويات القيمة الجديدة للمكدة في العنوان E21E وهي الرقم E1 تدفع إلى البأيت ذات القيمة العظمى من عداد البرنامج ، وتزداد قيمة محتويات مؤشر المكدة بمقدار واحد آخر فتصبح E21F وهي قمة المكدة (أو قاعها) التي كانت موجودة في بداية تنفيذ الأمر CALL E050 .

3. بذلك تصبح محتويات عداد البرنامج هي عنوان الأمر الموجود عند المكان E103 وهو الأمر الذي عليه الدور في التنفيذ في البرنامج الأساسي بعد أمر النداء مباشرة . إن هذه الخطوات قد تختلف اختلافاً بسيطاً من معالج لآخر ولكن يظل المعنى الأصلي كما هو .

في حالة البرامج الفرعية المعششة مع بعضها nested مهما كانت درجة تعبيشها فإنه عند كل أمر نداء CALL يتم تخزين عنوان الأمر التالي للأمر CALL مباشرة (أى الأمر الذي عليه الدور في التنفيذ) وهكذا إلى أن نصل إلى آخر برنامج فرعى في السلسلة حيث سيكون الأمر RET في آخره هو أول أمر RET يتم تنفيذه ونتيجة له يحمل عداد البرنامج بأول اثنين بآيت من قمة المكدة فيرجع التنفيذ إلى البرنامج الفرعى قبل الأخير وهكذا مع كل أمر RET يسحب عنوان (2 آيت) من قمة المكدة ويرجع التنفيذ إلى برنامج فرعى سابق إلى أن يصل التنفيذ إلى حيث انتهى من البرنامج الأساسي .

هناك أمر يمكنك من تخزين أى زوج مسجلات فى المكدة وهو الأمر :

PUSH rp

الذى يقوم بتنفيذ أول خطوتين فى شكل (4-12) ، مع ملاحظة أن محتويات عداد البرنامج لا تتغير هنا على الإطلاق لأنه ليس هناك أى قفر . كذلك فإن الأمر :

POP rp

يقوم بالعملية العكسية للأمر PUSH ، أى يسحب اثنين بآيت من المكدة ويضعهم فى زوج المسجلات المذكور فى الأمر POP وينقص محتويات مؤشر المكدة بمقدار اثنين . هذه العملية تكون مهمة جداً عند خدمة المقاطعة كما سترى في فصل قادم وهذا الأمر موجودان لجميع المعالجات التي ندرسها ويمكن مراجعتهما في قوائم الأوامر .

إن محتويات مؤشر المكدة التي تشير إلى قاعها أى أول مكان فيها يتم تحديدها عن طريق المبرمج في البرنامج باستخدام الأمر LXI SP,addr حيث يقوم هذا الأمر بتحميل المسجل SP بالعنوان addr الذي يختاره المبرمج ليكون عنوان قلع المكدة وذلك في المعالج 8085 ، أو LD SP,addr في حالة المعالج Z80 (لاحظ أنه قبل أى تعامل مع المكدة تكون قيمتها هي قاعها أى وهي فارغة) .

بذلك تكون قد أنهينا أهم ما يتعلق بالمكذبة ومؤشر المكذبة والبرامج الفرعية وفوائدها وننتقل الآن إلى بعض الأمثلة كتطبيقات على ذلك . لاحظ أن كل ما تم شرحه في هذا الباب يمكن تطبيقه على أي معالج من المعالجات التي درسناها (والتي سندرسها بخلاف طيف) ، فقط تكون حذرين عند كتابة شكل الأمر بالأسمى أو شفرته المستعشرية ، ولكن الأساس أو الهيكل العام لفكرة المكذبة والبرامج الفرعية التي شرحناها هي نفسها دائما .

4-12 حساب أزمنة التأخير

مثال 1-12

لدينا بوابة الإخراج رقم 00 وموصلاً عليها ثمانية دايودات أو موحدات ضوئية كل دايوه موصلاً على بث من بثات البوابة . المطلوب إتارة هذه الدايودات بالتتابع بحيث أن الدايوه الأول يضيء وبعده بنصف ثانية يضيء الدايوه الثاني وبعد بنصف ثانية أخرى يضيء الدايوه الثالث وهكذا حتى تصبح كل الدايودات مضيئة ، ثم بعد آخر دايوه بنصف ثانية تطفأ جميع الدايودات ثم يبدأ في الإضاءة من جديد بنفس الطريقة السابقة . اكتب برنامجاً يقوم بهذه المهمة مستخدماً البرامج الفرعية .

تعتمد فكرة البرنامج على عمل برنامج فرعي للتأخير بزمن مقداره نصف ثانية ثم تغادى على هذا البرنامج من البرنامج الأساسي بعد إنارة كل دايوه . قبل أن ندخل في تفاصيل البرنامج نريد أن نوضح هنا كيفية الحصول على أي زمان تأخير بأى قيمة . تقوم الفكرة أساساً على عمل حلقة ينفذها المعالج عدداً من المرات دون التأثير على أي شيء في البرنامج وبمعلومات عدد مرات تنفيذ هذه الحلقة وعدد الأوامر فيها وزمن تنفيذ كل أمر نستطيع حساب الزمن الكلى لتنفيذ الحلقة . انظر مثلاً لهذا البرنامج البسيط :

```
MVI C,FF; 7
LOOP: DCR C; 4
        JNZ LOOP; 10
```

الرقم الذى على يمين كل أمر هو عدد نبضات التزامن clock التي يأخذها الأمر حتى يتم تنفيذه وهذا العدد موضح فى قوائم أوامر كل واحد من المعالجات التي شرحنا طريقة برمجتها فى الفصول السابقة . فإذا كان تردد التزامن clock المستخدم هو 2 ميجا هرتز فإن ذلك يعني أن زمان كل نبضة هو نصف ميكروثانية . لذلك فإن زمان تنفيذ الأمر الأول مثلاً سيكون 3.5 ميكروثانية والأمر الثانى سينفذ فى 2 ميكروثانية والثالث سينفذ فى 5 ميكروثانية وهكذا . فى البرنامج السابق نلاحظ أن زمان تنفيذ الحالة مرة واحدة سيكون (4 +

(FF) 10 ميكروثانية ، وهذه الحلقة ستتكرر عدداً من المرات مقداره 256 $0.5 \times 7 = 3.5$ ميكروثانية ، وأما البرنامج السابق كلّه فسيعطي 1792 $1792 = 7 \times 256$ ميكروثانية . إذا كان هذا الزمن يكفي كتأخير لما تحتاج فقد انتهيت وإلا فعليك البحث عن طريقة تطيل بها هذا الزمن ، ومن ذلك استخدام زوج من المسجلات بدلاً من مسجل واحد كما في البرنامج التالي :

```

LXI D,FFFF; 10
LOOP: DCX D; 6
        MOV A,D; 4
        ORA E; 4
        JNZ LOOP; 10
    
```

هذا البرنامج يقوم بتحميل الزوج DE بالقيمة FFFF (65536) ثم يدخل في حلقة إنفصال بمقدار واحد إلى أن تصل محتويات زوج المسجلات إلى أصفاراً وعندما تنتهي الحلقة . لاحظ أن الأمر DCX ليس له تأثير على الأعلام لذلك فقد تم نقل محتويات مسجل من الآلتين إلى المركم ثم نفذت عملية OR على المسجلين والتي لن تكون نتيجتها صفراء إلا إذا كانت محتويات كل من المسجلين أصفاراً . زمن التأخير الذي سيعطيه هذا البرنامج سيكون :

$$= 786437 \text{ ميكروثانية} . \\ = 0.79 \text{ ثانية تقريباً} .$$

إذا كان هذا الزمن يفي بما تحتاج إليه من زمن تأخير فقد انتهيت ، وإلا فعليك استخدام الحلقات المعشقة كما في البرنامج التالي :

```

MVI B,FF; 7
LOOP1: MVI C,FF; 7
LOOP2: DCR C; 4
        JNZ LOOP2; 10
        DCR B; 4
        JNZ LOOP1; 10
    
```

$$\begin{aligned} \text{زمن تأخير الحلقة الداخلية} &= 256 \times 0.5 \times 14 = 1792 \text{ ميكروثانية تقريباً} . \\ \text{زمن تأخير الحلقة الخارجية} &= ((10+4+7) \times 256 + 1792) \times 0.5 = 461440 \text{ ميكروثانية تقريباً} . \end{aligned}$$

$$\begin{aligned} \text{زمن التأخير الكلي للبرنامج} &= 461440 + 3.5 = 461443.5 \text{ ميكروثانية} . \\ &= 0.46 \text{ ثانية تقريباً} . \end{aligned}$$

لاحظ أنه وإن كان زمن التأخير الناتج من حلقتين معشقتين أقل من زمن التأخير الناتج من زوج من المسجلات إلا أن الحلقات المعشقة يمكن تعويضها لأى درجة

فمثلاً في داخل الحلقة LOOP2 يمكن عمل حلقة ثالثة LOOP3 وهكذا للحصول على أزمنة تأخير كبيرة . أحياناً يكون المطلوب أزمنة محددة بقدر الإمكان كما في المثال الذي نحن بصدده الآن حيث المطلوب هو زمن تأخير مقداره 0.5 ثانية بالضبط . في هذه الحالة يمكن استخدام الأمر NOP في أماكن معينة في حلقات التأخير للضبط الدقيق للأزمنة المطلوبة . مثلاً ماذا سيكون الوضع لو أضفنا الأمر NOP داخل الحلقة الداخلية LOOP2 في البرنامج السابق ؟ وإجابة على ذلك فمن المعروف أن الأمر NOP يأخذ 4 نبضات تزامن لذلك سيصبح زمن التأخير الجديد هو 0.59 ثانية تقريباً ، وهذه القيمة أبعد من القيمة المطلوبة !! في هذه الحالة يمكن وضع الأمر NOP في الحلقة الخارجية ، وإذا لم يف (يفي) بالغرض المطلوب فيمكن تركه في الحلقة الداخلية مع التغيير في القيمة الابتدائية في أي من المسجلين A أو C ، ولقد وجدنا أنه بوضع القيمة الابتدائية D8 في المسجل B بدلاً من القيمة FF فإن زمن التأخير في هذه الحالة يساوي 499932 ميكروثانية وهي أقرب شيء إلى نصف الثانية . أي أن التعديل البسيط في القيمة النهاية لזמן التأخير ممكن بعدة طرق .

بعد أن رأينا كيفية الحصول على زمن التأخير المطلوب فإن البرنامج التالي يبيّن عملية الإضافة التتابعية ، وهذا البرنامج مكتوباً بلغة الأسsemblر للشريحة 8085 . هذا البرنامج كتب بصورة مبسطة جداً ويستطيع كل قارئ أن يأتي ببرنامج آخر يؤدي نفس الهدف وقد يكون أبسط من ذلك . ولقد تعمدنا كتابة البرنامج باستخدام العلامات LABELS حتى نترك للقارئ حرية كتابة البرنامج في أي مكان في الذاكرة يريد . بذلك تكون قد انتهينا من إعطاء القارئ فكرة كافية عن البرامج الفرعية وما يتعلق بها من المكدسة ومؤشر المكدسة .

```

START: MVI A,01
        OUT 00
        CALL DELAY
        MVI A,03
        OUT 00
        CALL DELAY
        MVI A,07
        OUT 00
        CALL DELAY
        MVI A,0F
        OUT 00
        CALL DELAY
        MVI A,1F
        OUT 00
        CALL DELAY
        MVI A,3F
    
```

```

OUT 00
CALL DELAY
MVI A,7F
OUT 00
CALL DELAY
MVI A,FF
OUT 00
CALL DELAY
MVI A,00
OUT 00
CALL DELAY
JMP START
DELAY: MVI B,DB
LOOP1: MVI C,FF
LOOP2: DCR C
NOP
JNZ LOOP2
DCR B
INZ LOOP1
RET

```

12-5 تمارين

1. شرح دور المكدة مع البرامج الفرعية ؟
2. ما هو الفرق بين القفز العادى فى أى برنامج والقفز إلى برنامج فرعى ؟
3. اشرح دور الأمر RET (أمر العودة من البرنامج الفرعية) فى ضمان عودة المعالج إلى نفس المكان الذى خرج منه فى البرنامج الأساسي ؟
4. لماذا يكون من الضرورى عادة استخدام الأمر PUSH فى أول البرنامج الفرعى والأمر POP فى آخره ؟
5. يحتوى هذا الفصل على بعض البرامج الفرعية للحصول على أزمنة التأخير ، فهل يتغير مقدار هذه الأزمنة بتغيير التزامن clock المستخدمة ؟
6. اكتب برنامجا فرعيا للحصول على زمن تأخير مقداره 100مليثانية مع العلم أن التزامن يساوى 2 ميجا هرتز ؟
7. اكتب برنامجا فرعيا آخر يستخدم البرنامج الفرعى السابق للحصول على زمن تأخير مقداره ثانية واحدة ؟
8. اكتب برنامجا فرعيا آخر يستخدم البرنامجين السابقين للحصول على زمن تأخير مقداره دقيقة واحدة ؟

9. استخدم البرامج الفرعية السابقة فى عمل ساعة رقمية تظهر الثوانى والدقائق وال ساعات على مظهرات السبع قطع 7 segment ؟
10. اكتب برنامج التحكم فى إشارات المرور فى الفصل السابق مستخدما البرامج الفرعية ولاحظ الفرق ؟
11. اكتب برنامج يحسب قيمة X التالية :

$$X = 5! + 8! + 9!$$

حيث ! تمثل مضروب الرقم .

12. اكتب برنامج يحسب قيمة X التالية :

$$X = 5^5 + 8^4 + 9^3$$

الفصل الثالث عشر

المقاطعة

Interrupt

1-13 مقدمة

لقد رأينا في الفصول السابقة كيفية مواجهة المعالج مع الأجهزة المحيطة سواء كانت ذاكرة أو بوابات إدخال وإخراج . في العادة يقوم المبرمج بكتابية برنامج على أساسه يقوم المعالج بخدمة هذه الأجهزة المحيطة . هناك طريقتان يمكن للمعالج أن يخدم بهما هذه الأجهزة المحيطة وهما :

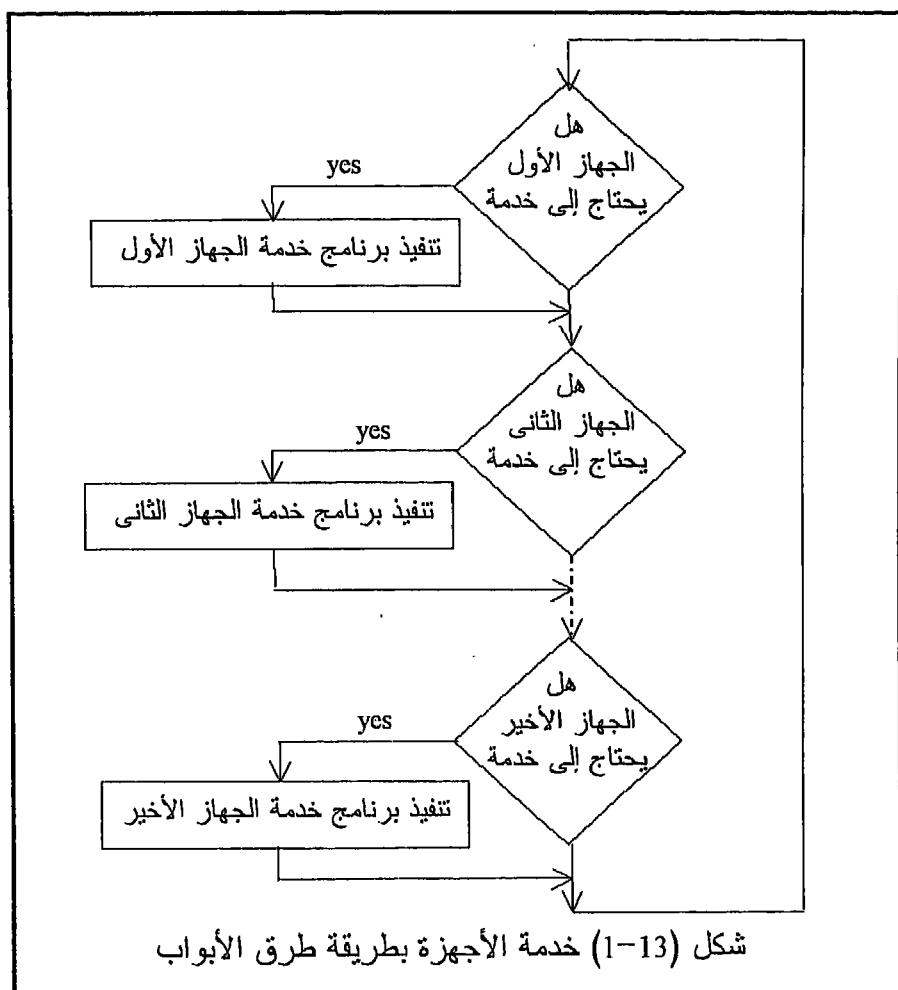
1. الطريقة الأولى سنسميها "خدمة طرق الأبواب" التي عرفت في المراجع الإنجليزية بكلمة polling والتي لها نفس المعنى كما سنرى ولكن الترجمة ليست حرفية .
2. الطريقة الثانية هي طريقة المقاطعة interrupt وهي الطريقة التي سنشرحها بالتفصيل في هذا الفصل ، ولكن لا مانع من القاء نظرة سريعة على الطريقة الأولى حتى يختار القارئ أى الطريقتين يجب أن يستخدم .

2- طريقة طرق الأبواب لخدمة الأجهزة المحيطة Polling service

تعتمد هذه الطريقة على أن المعالج يقوم بطرق أبواب جميع الأجهزة المحيطة بالتتابع ويسأل كل منها هل تزيد خدمة أقوم بأدائها ؟ فإذا كانت إجابة الجهاز بنعم فإن المعالج يقوم بتنفيذ هذه الخدمة له وفي الحال دون أي انتظار ، أما إذا كانت إجابة الجهاز بالنفي فإن المعالج ينتقل إلى الجهاز التالي له ويسأله نفس السؤال ، وهكذا إلى أن يصل المعالج إلى آخر جهاز فإذا أدى خدماته أو لا على حسب إجابته . بعد آخر جهاز يرجع المعالج إلى أول جهاز ويعيد الكرة من جديد إلى مالاينهاية . أى أن الوظيفة الوحيدة للمعالجة في هذه الحالة هي الدوران إلى مالاينهاية على الأجهزة المحيطة وتقديم الخدمة لمن يجيئه . شكل (13-1) يبين مخطط سير لوظيفة المعالج في هذه الحالة . لقد استخدمنا هذه الطريقة إلى حد ما في مثل إشارات المرور الذي سبق شرحه في الفصل الثاني عشر حيث كان المعالج دائماً يسأل بوابة الإدخال ، أى يقرأها ، فإذا كانت واحداً يقوم بتنفيذ برنامج معين ، وإذا كانت صفرًا يقوم بتنفيذ برنامج آخر وبعد الانتهاء من أى من البرنامجين يرجع ويقرأ بوابة الإدخال مرة ثانية وهكذا إلى مالاينهاية .

إن من مميزات طريقة طرق الأبواب لخدمة الأجهزة المحيطة أنها سهلة البرمجة ولا تحتاج إلى الكثير من التجهيزات hardware . وإن من عيوبها أن المعالج يكون مخصصاً لوظيفة خدمة هذه الأجهزة ولا يستطيع الفكاك منها وأنت كمبرمجة لا تستطيع عادة الاستفادة منه في أى أغراض أخرى ، وبالذات إذا كان عدد

الأجهزة التي يقوم المعالج بالمرور عليها قليلاً فإن ذلك يعتبر إهداراً لفعالية المعالج . أما إذا كان عدد الأجهزة التي يقوم المعالج بخدمتها بهذه الطريقة كبيراً فإن المعالج قد يتاخر على بعض الأجهزة التي تحتاج لخدمته على فترات متقاربة لأن عليها الانتظار لحين أن يجيء دورها ، كما أنها ليس من حقها أن تقطّع المعالج وتطلب الخدمة الفورية في حالات الضرورة ، وكما نعلم فإن هناك الكثير من المواقف التي تستلزم الخدمة من المعالج فوراً كضرب جرس إنذار مثلاً عند حدوث أي طارئ في أي نظام تحكم مثل إنقطاع الكهرباء أو حريق أو إرتفاع زائد في ضغط غاز وغيرها الكثير . لذلك فإن المعالج يقدم للمبرمج نوعاً آخر من الخدمة وهي المقاطعة التي سننكلم عنها بالتفصيل في الجزء القادم .



Interrupt 3- المقاطعة

في نظام خدمة الأجهزة الخارجية عن طريق المقاطعة لا يذهب المعالج إلى الأجهزة ويطرق بابها ليعرض عليها خدماته فإن أردت أعطى وإن أبت يذهب إلى جهاز آخر ، لا ، بل إن المعالج هنا يكون عادة مشغولاً في تنفيذ برنامج معين وعادة ما يكون هذا البرنامج لا نهائى فإذا احتاج أحد الأجهزة لخدمة من المعالج فإنه يقاطعه ويطلب منه الخدمة فيقوم المعالج بتنفيذ هذه الخدمة للجهاز المقاطع وبعد الانتهاء من هذه الخدمة يعود المعالج لتنفيذ البرنامج الأساسي من حيث انتهى قبل المقاطعة . إن ذلك تماماً أنك تكون واقعاً مع أحد زملائك تتحدثان ، ثم يجيء آخر ويقاطعكم ليسأل عن موعد محاضرة المعالجات ، فـى هذه الحالة يقطع المتكلم محادثته ليجيب السائل عن طلبه ثم بعد الإجابة يستأنف حديثه الأول . إن البرنامج الذي يقوم المعالج بتنفيذه عند المقاطعة يسمى برنامج خدمة المقاطعة .

من مميزات هذه الطريقة أن الأجهزة المقاطعة تستطيع مقاطعة المعالج في أي وقت تريده وليس عليها الانتظار إلى دورها مثل الطريقة السابقة ، وإذا تصادف وتمت المقاطعة في نفس الوقت من أكثر من جهاز فإن المعالج يخدمها حسب أولويات تحدد له من قبل المستخدم مسبقاً ، فكيف يتم ذلك ؟ في أثناء اشغال المعالج في تنفيذ برنامج خدمة مقاطعة معينة ، هل يستطيع جهاز آخر أن يقاطعه؟ كيف نكتب برنامج خدمة المقاطعة ؟ وكيف يرجع المعالج إلى نفس مكانه في البرنامج الأساسي بعد إنتهاء خدمة المقاطعة ؟ كل هذه الأسئلة وأكثر سنجيب عليها من خلال دراستنا للأجزاء القادمة وبعد دراستنا لتفاصيل مقاطعة كل معالج من المعالجات التي درسناها .

إن المقاطعة تكون عادة عن طريق إشارة يرسلها الجهاز المقاطع إلى المعالج على أحد أطرافه المخصصة لذلك وعندما يكتشف المعالج هذه الإشارة فإنه يقوم بتنفيذ برنامج خدمة المقاطعة وذلك يرجع لأن المعالج يقوم بقراءة جميع أطراف المقاطعة وإختبارها قبل الدخول في تنفيذ أي أمر . وإليك بعض الأمثلة التي تستخدم فيها المقاطعة :

- الأجهزة الخارجية مثل الطابعة ولوحة المفاتيح يمكنها أن تقاطع المعالج وترسل أو تستقبل أي معلومات .
- يمكن في أي وقت مقاطعة أي برنامج يتم تنفيذه إذا كان هذا البرنامج ينفذ بطريقة خطأ .
- يمكن للعمليات الصناعية التي يتم مراقبتها باستخدام المعالج أن تقاطعة في أي لحظة طوارئ تحدث للعملية الصناعية .

- عند إعطاء إشارة المقاطعة لأى معالج من المعالجات التى نقوم بدراستها يحدث الآتى مع بعض الاختلافات البسيطة من معالج لأخر سنشير إليها فى موضعها .
1. الأمر الحالى يتم إكمال تنفيذه من قبل المعالج .
 2. عنوان الأمر الذى عليه الدور فى التنفيذ (محتويات عداد البرنامج) تخزن فى المكادسة Stack حتى يمكن العودة إليه عند الانتهاء من خدمة المقاطعة . كما يتم تخزين محتويات أى مسجل يخشى من تغيير محتوياته فى أثناء خدمة المقاطعة ويتم ذلك عن طريق المبرمج .
 3. كل إشارة مقاطعة لها عنوان خاص مصاحب لها كما سنرى ، يتم وضع هذا العنوان فى عداد البرنامج (عن طريق المعالج) حيث يقف المعالج إلى هذا العنوان ويبداً فى تنفيذ البرنامج الذى يكون هذا هو عنوان أول أمر فيه ويسمى هذا البرنامج ببرنامج خدمة المقاطعة ونتم كتابته عن طريق المستخدم .
 4. بعد الانتهاء من برنامج خدمة المقاطعة يعود المعالج إلى البرنامج الأصلى ليستأنف تنفيذه من نفس مكان المقاطعة بالاستعانة بالعنوان الذى تم تخزينه فى المكادسة كما فى الخطوة رقم 2 .

إن الجهاز المقاطع فى الكثير من الأحيان وبعد إعطاء إشارة المقاطعة وقبول المعالج لها والبدء فى برنامج الخدمة فإن الباب يظل مفتوحاً للأجهزة الأخرى المقاطعة مما قد ينشأ عنه موقف يجب الحرص منه . لو أن المعالج بدأ فى خدمة المقاطعة وما زال طرف المقاطعة الموصل على الجهاز المقاطع فعلاً فإن المعالج سيبداً فى خدمة نفس المقاطعة من جديد على الرغم من أنه لم ينته من الخدمة السابقة ، وب مجرد أن يبدأ فى خدمة المقاطعة من جديد للمرة الثانية وما زال خط المقاطعة فعلاً من قبل الجهاز المقاطع فإن المعالج سيبداً فى الخدمة من جديد أيضاً ، وهكذا فإن المعالج سيدخل فى حلقة لا نهاية لـ لن يخرج منها دون أن ينفذ برنامج خدمة المقاطعة . لتجنب هذا الموقف يجب على المبرمج أن يضع أمراً معيناً فى بداية برنامج خدمة المقاطعة يمنع المعالج من خدمة أى مقاطعة إلى أن ينتهى من الخدمة الحالية التى دخل فيها . إن هذه العملية لها تفاصيل ستختلف من معالج إلى آخر لذلك سنرجىء الكلام عنها الآن .

8085-4 مقاطعة المعالج

شكل (2-13) يبين جدول لجميع أطراف المقاطعة الخاصة بالشريحة 8085 وعنوان المكان الذى يتم القفز إليه عند إعطاء إشارة المقاطعة على هذا الطرف وكذلك الأولوية الخاصة بكل طرف من أطراف المقاطعة وكيفية إعطاء هذه الإشارة على كل طرف . كما نلاحظ من هذا الشكل فإن الطرف TRAP له أعلى أولوية ثم يليه الطرف RST7.5 فالطرف RST6.5 ثم الطرف RST5.5 ثم يكون

الطرف INTR له أقل أولوية . إن كلمة أولوية هنا تعنى أنه إذا جاءت إشاراتان على خطين مختلفين من خطوط المقاطعة في نفس الوقت تماما فإن الإشارة التي على الخط ذى الأولوية الأعلى هي التي تؤخذ فى الاعتبار أما الإشارة الأخرى فتُهمل .

طريق المقاطعة	الأولوية	العنوان الذى يتم القراءة إليه	كيفية إعطاء الإشارة على الطرف
TRAP	1	0024H	الحافة صفر إلى واحد وبقى واحد إلى أن تقبل المقاطعة
RST7.5	2	003CH	الحافة صفر إلى واحد حيث يتسم مسک هذا الواحد في المعالج
RST6.5	3	0034H	يبقى واحد إلى أن تقبل المقاطعة
RST5.5	4	002CH	يبقى واحد إلى أن تقبل المقاطعة
INTR	5	له تفاصيل	يبقى واحد إلى أن تقبل المقاطعة

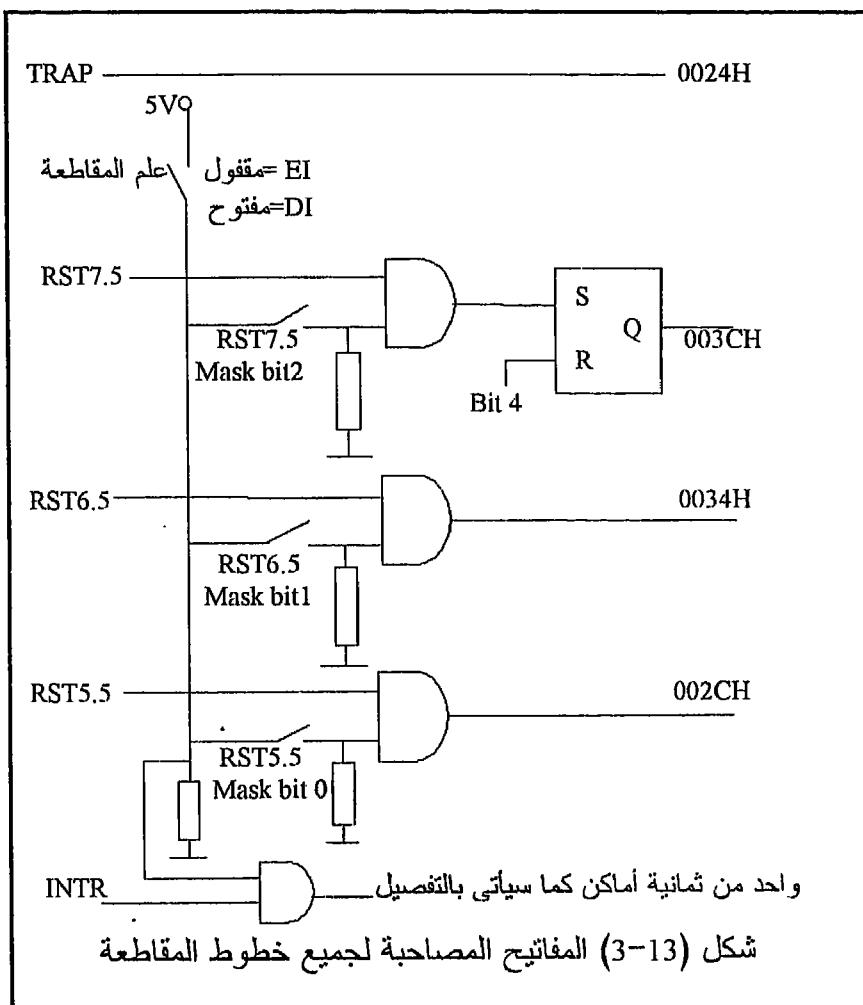
شكل (13-2) أطراف المقاطعة للشريحة 8085 وأولوياتها وعنوان القراءة الخاصة بكل منها وكيفية إعطاء كل إشارة على هذه الأطراف

13-4-1 الخطوط RST5.5, RST6.5, RST7.5

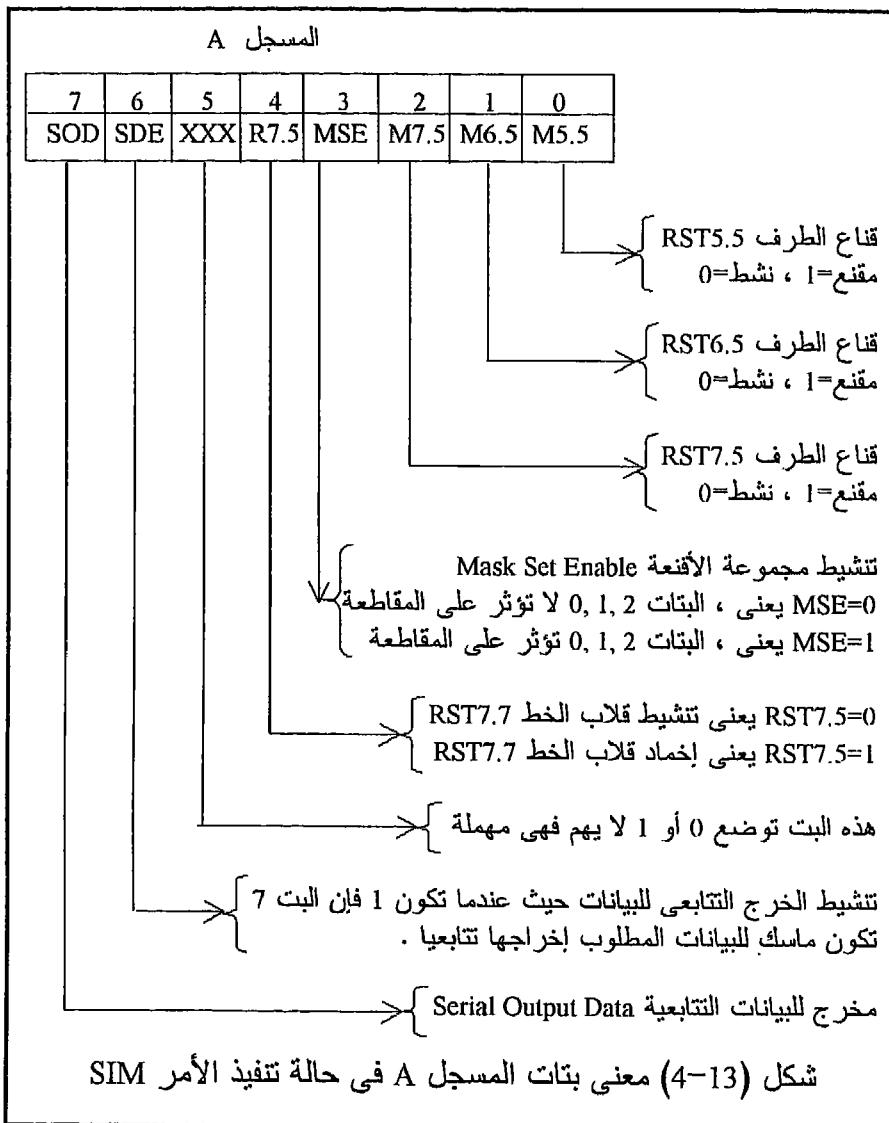
بالنسبة للمعالج 8085 هناك بعض الأوامر المتعلقة بالمقاطعة والتي يجب معرفتها أو لا ومن هذه الأوامر ما يلى :

- الأمر EI ومعناه تنشيط المقاطعة Enable Interrupt . هناك في داخل شريحة المعالج قلابا flip flop يسمى قلاب تنشيط المقاطعة ، Interrupt Enable flip flop ، أى أن أي إشارة مقاطعة على جميع الخطوط (RST5.5, RST6.5, RST7.5, (INTR) ما عدا الخط TRAP لن تنفذ إلا إذا كان خرج هذا القلاب يساوى واحدا . لذلك فإن هذا القلاب يعتبر بمثابة مفتاح مركب على التوالي مع هذه الخطوط مجتمعه يتم فتحه (ON) بالأمر EI . شكل (13-3) يبين وضع هذا المفتاح بالنسبة لهذه الخطوط .
- الأمر DI ومعناه إخمد أو امنع أو لا تقبل المقاطعة Dissable Interrupt وكما نرى فإن تأثيره على قلاب تنشيط المقاطعة يكون تماما عكس تأثير الأمر EI . يستخدم الأمر DI في حجب أو إخماد المقاطعة في بعض أجزاء من البرنامج ومن أهمها برامج خدمة المقاطعة نفسها . لذلك فإنه يجب على المبرمج أن يضع الأمر DI في بداية أي برنامج لخدمة المقاطعة لحماية هذه المقاطعة من نفسها ومن الدخول في الحلقة اللاهائية التي ذكرناها منذ قليل . في نهاية برنامج خدمة المقاطعة يضع المبرمج الأمر EI لتنشيط المقاطعة من جديد .

- مثلاً أن هناك مفتاحاً عمومياً (علم المقاطعة) لجميع أطراف المقاطعة ما عدا الخط TRAP يمكن تنشيطها به أو حجبها به عن طريق الأمر EI و DI فـإن هناك لكل واحد من الخطوط RST7.5 و RST6.5 و RST5.5 مفتاحاً خاصاً به يمكن به تنشيط هذا الخط أو حجبه ، كما أن الخط RST7.5 له مفتاح أو قلاب آخر إضافي خاص به هو فقط يمكن منه حجب المقاطعة عليه حتى لو مررت من خلال المفتاح الأول وهذه ميزة للخط RST7.5 عن باقي الخطوط الأخرى . انظر لهذه القلابات أو المفاتيح في شكل (3-13) . هذه المفاتيح يمكن تنشيطها أو حجبها عن طريق الأمر SIM والذي يعني "ضع أقنية المقاطعة" Set Interrupt . Masks



إن هذا الأمر عند تطبيقه يتم تنشيط أو حجب أو وضع قناع على أي خط من هذه الخطوط على ضوء شفرة توضع في مسجل التراكم قبل هذا الأمر مباشرة . شكل (4-13) يبين علاقة بิตات مسجل التراكم مع هذه المفاتيح . نلاحظ من هذا الشكل مثلاً أن قناع الخط RST5.5 هو البت رقم 0 فإذا كانت هذه البت تساوى صفرًا فإن الخط RST5.5 يكون نشطاً أى غير مقنع أو غير محظوظ ، أمّا إذا كانت هذه البت تساوى واحداً فـإن ذلك يعني أن الخط RST5.5 يكون مقنعاً أو محظوظاً .



نفس الشيء يمكن أن يقال عن الخطوط RST6.5 و RST7.5 و بتاتها المقابلة رقمي واحد وإنفين . البت D4 تمثل المفتاح الإضافي الخاص بالخط RST7.5 فإذا كانت هذه البت تساوى واحد فإن المقاطعه RST7.5 تكون مقنعة أو محجوبة . شكل (3-13) يبين رسمياً توضيحاً لمفاتيح الأقنعة لكل خط مقاطعة وكيفية التحكم بهذه المفاتيح وعلاقتها ببعضها .

مثال 1-13

اكتب الأمر الذى ينشط المقاطعة RST5.5 و RST7.5 ويحجب المقاطعة RST6.5 . طالما أن RST5.5 نشط ، إذن $D0=0$ وكذلك طالما أن RST7.5 نشط $D1=1$ إذن $D2=0$ وأيضاً $D4=0$ وطالما أن RST6.5 مقطوع أو محجوب إذن $D3=1$ والحصول على ذلك الوضع لابد وأن تكون $D6=0$. وأما $D6$ فلا بد أن تكون صفراء لأننا لسنا في حالة إخراج بيانات تتبعية وعلى ذلك فإن البت $D7$ لا يهم في هذه الحالة أن تكون صفراء أو واحداً طالما أن $D6=0$. كذلك فإن $D5$ لا يهم أن تكون صفراء أو واحداً . وعلى ذلك فإن محتويات المركم ستكون كما يلى :

D7 D6 D5 D4 D3 D2 D1 D0
0 0 0 0 1 0 1 0 = 0AH

ولكي يتم ذلك ننفذ الأمرين التاليين :

MVI A,0AH

SIM

مثال 2-13

تخيل أن هناك جهاز إستقبال للبيانات التتابعية موصلاً على الطرف SOD (طرف رقم 4) للشريحة 8085 ، والمطلوب هو إرسال 0 إلى هذا الجهاز كتتشيط لكي يبدأ في الاستقبال وذلك دون التأثير على حالة أطراف المقاطعة التي تم تجهيزها في المثال الماضي .

من شكل (2-13) نجد أنه حتى نترك أقنعة المقاطعة على البتات D0,D1,D2,D4 كما هي يجب أن تكون $D3=0$ ، وحتى ننشط البت D7 كبت للإخراج التتابعى فإن $D6=1$ ، ونضع $D7=0$ لكي نخرجه على الطرف SOD . لاحظ أن البتات D0,D1,D2,D4 في هذه الحالة لا يهم أن تكون صفراء أو واحداً وسنفترضها أصفاراً ، وعلى ذلك تكون محتويات المركم كالتالى :

D7 D6 D5 D4 D3 D2 D1 D0

0 1 0 0 0 0 0 0 = 40H

ولكي يتم ذلك ننفذ الأمرين التاليين :

MVI A,40H

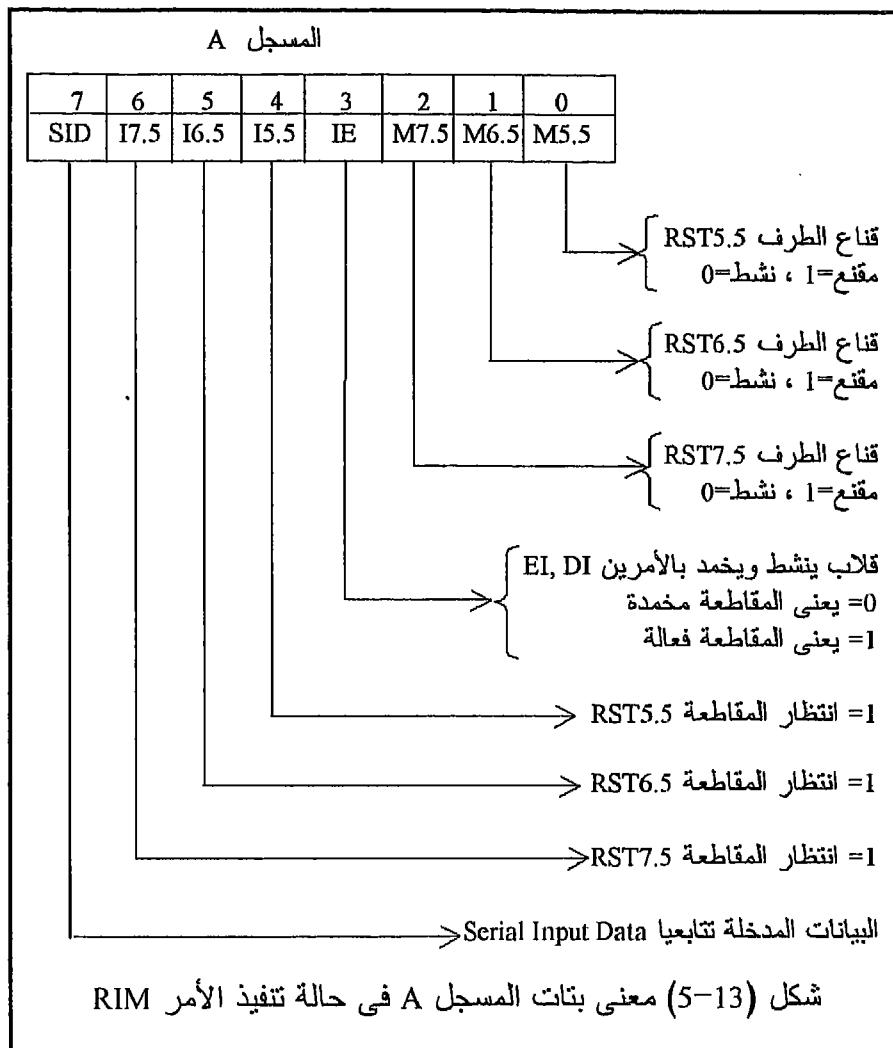
SIM

لاحظ أن الخط TRAP لا يتاثر بأى شيء من ذلك .

يمكن عمل مقارنة بين خطوط المقاطعة RST5.5 و RST6.5 و RST7.5 و ثلاثة خطوط تليفونات تخرج من سنترال فى شركة معينة كالتالى: تخيل أن واحداً من هذه الخطوط هو تليفون رئيس الشركة وهو الخط RST7.5 والخط الثاني هو خط نائب الرئيس وهو الخط RST6.5 وأما الخط الثالث فهو خط مدير شئون الأفراد وهو الخط RST5.5 . هناك مكتب استقبال يرد على التليفونات القادمة للأشخاص الثلاثة ويوجهها إلى الشخص المطلوب . فى وقت الدوام (العمل) يعمل سنترال الشركة ويكون نشطاً وفي غير وقت الدوام لا يعمل هذا السنترال ، ذلك يكفىء تماماً تنشيط هذه الخطوط بالأمر EI وحجبها أو إخمادها بالخط DI . كل واحد من الأشخاص الثلاثة (الرئيس ونائبه ومدير شئون الأفراد) يترك خبراً في مكتب الاستقبال عما إذا كان يريد استقبال مكالمات أم لا ، ذلك يكفىء وضع الشفرة المناسبة في البتات 0 إلى 2 في مسجل التراكم ثم تتفيد الأمر SIM .
إفترض أن السنترال يعمل والاستقبال موجود أيضاً فإنه سيرد على المكالمات ويوصلها أو يمنعها عن الأشخاص الثلاثة وذلك على حسب الأوامر التي ترکوها عنه ، إن ذلك يكفىء تماماً كون البت 3 تساوى واحداً في الشفرة الموجودة في المسجل A . أحياناً يكون السنترال يعمل ولكن الاستقبال مشغول ، في هذه الحالة كل المكالمات القادمة ستوصلك أو لا توصلك على حسب آخر قائمة أوامر من الأشخاص الثلاثة دون أي اعتبار لأى تغيير يريده أى واحد منهم وهذا يكفىء حالة المسجل A قبل آخر مرة ينفذ فيها الأمر SIM وتكون البت 3 تساوى صفراء في هذه الحالة . هناك ميزة فريدة للرئيس وهي أن عنده مفتاحاً خاصاً به بحيث عندما يكون هذا المفتاح ON ترد ماكينة إجابة ذاتية تقول "المدير مشغول الآن من فضلك اتصل مرة أخرى" . هذا يكفىء تماماً البت 4 التي عندما تكون صفراء فإن الخط RST7.5 يكون فعالاً وإذا كانت واحداً فإن هذا الخط يكون مقنعاً أو محظياً .

هناك الأمر RIM الذي يتكون من بaitt واحدة والذي يقوم تقريباً بالعملية العكسية للأمر SIM حيث يقوم بتحميل المسجل A بثمانية بتات توضح حالة أقنية المقاطعة . لذلك فإن هذا الأمر معناه "اقرأ أقنية المقاطعة" Read Interrupt Masks . في حالة قراءة أقنية المقاطعة بالأمر RIM فإن بتات المسجل A تترجم محتوياتها كما في شكل (5-13) حيث الثلاثة بت الأولى 0 إلى 2 هي حالة أقنية خطوط المقاطعة RST5.5 و RST6.5 و RST7.5 كما تم تسجيلها باستخدام الأمر SIM مسبقاً . البت الثالثة تبين حالة قلاب أو علم المقاطعة الذي رأينا كيف نتحكم فيه بالأمرتين EI و DI . افترض أن المعالج يقوم الآن بخدمة مقاطعة للخط RST7.5 ، وفي أثناء ذلك افترض أن الخط RST6.5 طلب المقاطعة ، ماذا سيفعل المعالج ؟ إن المعالج يضع الخط RST6.5 في حالة إنتظار

إلى حين الانتهاء من خدمة المقاطعة RST7.5 . في هذه الحالة يستطيع المبرمج كتابة بعض الأوامر في آخر برنامج خدمة مقاطعة الطرف RST7.5 يجعل المعالج يذهب إلى خدمة المقاطعة RST6.5 بدلًا من العودة للبرنامج الأساسي . البت رقم 5 في مسجل التراكم تعكس هذه الحالة ، فإذا كان الطرف RST6.5 في حالة انتظار فإن هذه البت تكون واحدا وتكون صفرًا في غير ذلك . البت رقم 4 تمثل الانتظار لطرف المقاطعة RST5.5 والبت رقم 6 تمثل الانتظار لطرف المقاطعة RST7.5 . البت الأخيرة في مسجل التراكم كما في شكل (13-5) تمثل البيانات المدخلة تتبعياً إن وجدت حيث سنترك الحديث عن الإدخال والإخراج التابعى للبيانات الآن .



شكل (6-13) يبين مثلا على اختبار المقاطعة RST6.5 في نهاية برنامج مقاطعة الخط RST7.5 بحيث إذا كانت في حالة انتظار يذهب المعالج لخدمتها قبل الرجوع للبرنامج الأساسي .

		نهاية برنامج خدمة مقاطعة الخط RST7.5 ;
RIM		قراءة الأقنية المقاطعة ;
MOV B,A		تخزين معلومات الأقنية في المسجل B ;
ANI 20H		اختبار لمعرفة إذا كان الخط RST6.5
		في الانتظار ، أي البت خمسة من A تساوى واحد ؛
JNZ NEXT		
EI		لا ينتظر ، نشط المقاطعة ؛
RET		عودة للبرنامج الأساسي ؛
NEXT: MOV A,B	A	أعد حالة الأقنية للمسجل A
ANI 0DH		تنشيط المقاطعة RST6.5 قد تكون مقنعة ؛
ORI 08H		تنشيط البتات 0 إلى 2 قد تكون خاملة ؛
SIM		سجل الصورة الجديدة للأقنية ؛
JMP SERV6.5		إفز إلى برنامج خدمة المقاطعة RST6.5 ؛
شكل (6-13) خدمة RST6.5 من برنامج خدمة RST7.5		

مثال 3-13

افترض أنه بعد تنفيذ الأمر RIM وجدت المحتويات التالية في المسجل A :
D7 D6 D5 D4 D3 D2 D1 D0
0 1 0 0 1 0 0 1 = 49H

إن ذلك يعني أن المقاطعة RST5.5 هي المحجوبة (D0=1) وأما المقاطعة RST6.5 و RST7.5 فنشيطان (D2D1=00) ، كذلك فإن علم المقاطعة IF نشيط أي أن الأمر EI مازال سارى المفعول ، والمقاطعة RST7.5 فقط هي التي فى حالة انتظار (D7=0) كما أن هناك 0 قادم من الطرف SID (D6D5D4=100) .

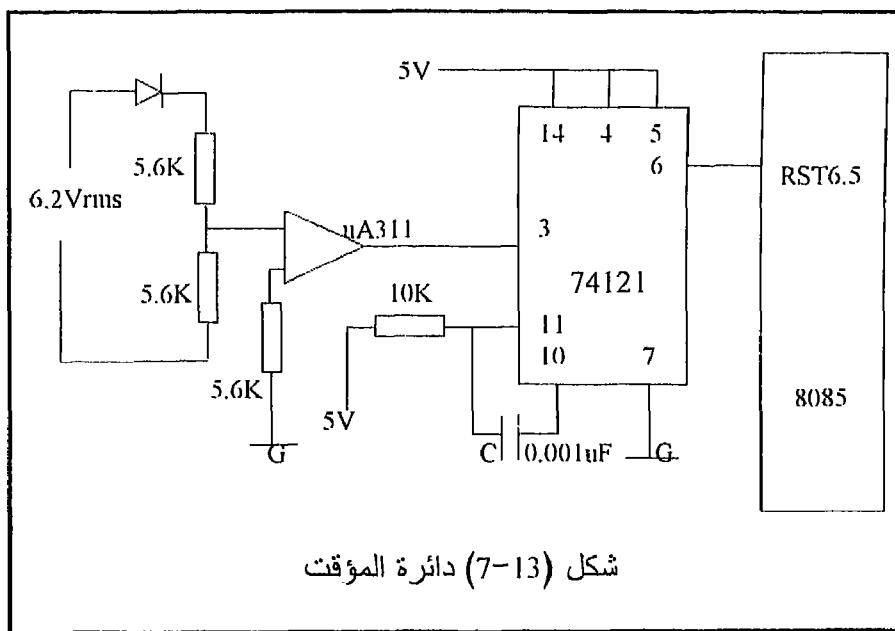
مثال 4-13

المطلوب عمل مؤقت timer لزمن مقداره دقة واحدة باستخدام التردد 60 هرتز الناتج من خط القدرة . هذا المؤقت يعرض الزمن على 3 خانات واحدة لل دقائق وإثنان للثوانى بحيث تعرض خانة الدقائق واحدا (أى دقة واحدة) ثم يبدأ العد

التصاعدى للثانى إلى أن تصل إلى 60 فترجع إلى الصفر مرة أخرى وتستمر خاتمة الدقائق تعرض واحداً وتبداً الثانى في العد التصاعدى ، وهكذا .

شكل (7-13) يبين الدائرة المستخدمة لهذا الغرض وشكل (7-8) يبين البرنامج المقترن لذلك . تعتمد الدائرة المستخدمة على تحويل جهد القدرة من 110 فولت إلى 6 فولت باستخدام محول خافض للجهد ثم تقسيم هذا الجهد وإدخاله على مقارن mA311 ليحول الموجة الجيبية إلى موجة مربعة TTL . تستخدم هذه الموجة لعمل إثارة trigger للشريحة 74121 التي تخرج نبضة TTL يمكن التحكم في عرضها باستخدام المكثف C والمقاومة R . هذه النبضة الخارجة من الشريحة 74121 توصل على الطرف RST6.5 حيث ستعطى 60 نبضة مقاطعة في الثانية الواحدة على هذا الطرف .

لكى نفهم البرنامج سنسير معه بالخطوات التالية :



شكل (7-13) دائرة المؤقت

أولاً: عندما ستجيء نبضة مقاطعة على الطرف RST6.5 من الشريحة 74121 فإن المعالج سيقفز إلى العنوان 0034H كما أوضحنا في شكل (2-13) . هذا العنوان يقع في أول صفحة من الذاكرة حيث كل صفحة من الذاكرة تحتوى 256 بait (FFH) وعادة ما يكون هذا الجزء من الذاكرة مشغولاً بـ EPROM تحتوى بعض البرامج الخاصة بتشغيل الميكروكمبيوتر . إذا كان الأمر كذلك فإن مبرمج الـ EPROM عادة يسجل في مثل هذه الأماكن أمر قفز إلى مكان آخر بحيث يكون هذا المكان في الـ RAM . لذلك يجب عليك أن تقرأ كتالوج الجهاز

الذى تتدريب عليه وتعرف منه أمر القفز الموجود عند العنوان 0034 أين يقفز فى ال RAM . سنفترض أن هذا العنوان هو أول عنوان وسنعطيه الرمز xxxx حيث سنكتب عند هذا العنوان برنامج خدمة المقاطعة .

0034 JMP xxxx ;	----- هذا هو البرنامج الأساسى -----
LXI SP,STCK ; STCK	تحديد بداية المكادسة بالعنوان STCK ;
MVI A,1DH	الرقم 1D ينشط المقاطعة RST6.5 ويحمد الباقي ;
SIM	تنشيط المقاطعة RST6.5 ;
LXI B,0000H	تصفيير المسجلين C, B للدقائق والثوانى ;
MVI D,3CH	الرقم 3CH يعادل 60 وهى عدد الثوانى ;
EI	تنشيط المقاطعة قبل الدخول فى برنامج إظهار الأرقام ;
DSPLY: MOV A,B	
OUT 00	إظهار الدقائق على البوابة 00 ;
MOV A,C	
OUT 01	إظهار الثوانى على البوابة 01 .
JMP DSPLY	
xxxx: DCR D	بداية برنامج خدمة المقاطعة فى ال RAM ;
EI	تنشيط المقاطعة قبل الرجوع للبرنامج الأساسى ;
RNZ	عودة طالما أن عدد المقاطعات لم يصل إلى 60 ;
DI	إخماد المقاطعة إذا وصل العدد إلى 60 مقاطعة ;
MVI D,3CH	تحميل المسجل D بالرقم 60 ثانية مرة أخرى ;
INC C	زد الثوانى بمقدار واحد ;
MOV A,C	
DAA	وضع الثوانى فى الصورة العشرية ;
MOV C,A	
CPI 60	هل وصل عدد الثوانى إلى 60 ثانية ;
EI	تنشيط المقاطعة قبل العودة للبرنامج الأساسى ;
RNZ	عودة طالما لم نصل إلى 60 ثانية ;
DI	إخماد المقاطعة إذا وصل عدد الثوانى إلى 60 ثانية ;
MVI C,00	تصفيير عداد الثوانى ;
MVI B,01	حمل المسجل B بدقة (زمن التأخير) ;
EI;	تنشيط المقاطعة قبل العودة ;
RET ;	عودة للبرنامج الأصلى .

شكل (8-13) برنامج المؤقت باستخدام المقاطعة

ثانياً : أهم جزء في البرنامج الأساسي هو حلقة لا نهائية تخرج على بوابتي الإخراج 00 و 01 قيمة الثنائي التي تتغير كل ثانية وقيمة الدقائق التي هي واحد باستمرار والتي يمكن التحكم فيها من البرنامج ، البوابات غير مبينة في شكل (13-7) للتبسيط فقط .

ثالثاً : مع كل نبضة مقاطعة على الطرف RST6.5 سيقفز المعالج من الحلقة الالاتهائية إلى العنوان 0034 ومنه إلى العنوان xxxx حيث سيبدأ من هناك تنفيذ برنامج خدمة المقاطعة .

رابعاً : في برنامج خدمة المقاطعة سيكون هناك عدد تزداد محتوياته بمقدار واحد مع كل نبضة مقاطعة بحيث أنه طالما أن محتويات هذا العدد لم تصل إلى 60 فإن المعالج يرجع إلى البرنامج الأساسي في الحلقة الالاتهائية حيث يستأنف عمله كما في الخطوة (ثانياً) .

خامساً : مع تكرار المقاطعة يصل العدد في برنامج الخدمة إلى 60 حيث عندها يقوم برنامج الخدمة بتعديل قراءة الثنائي بزيادتها بمقدار ثانية واحدة .

سادساً : إذا وصل عدد الثنائي إلى 60 يكون قد مضى دقيقة ، عندها يرجع عدد الثنائي إلى الصفر لتبدأ العملية من جديد .

شكل (13-2) يبيّن كيفية استقبال المعالج لنبضات المقاطعة على الخطوط RST5.5, RST6.5, RST7.5 حيث نلاحظ من هذا الشكل أنه بالنسبة للخط RST7.5 بالذات يكفي أن تصعد النبضة من صفر low إلى واحد high لكي يستقبل المعالج هذه الإشارة وذلك لوجود ماسك في مدخل هذا الخط ، الخطان RST5.5, RST6.5 لا بد أن تبقيا واحدا high إلى أن يقبلها المعالج وإلا لو أنها نزلت إلى الصفر low قبل أن يقبلها المعالج فإن هذه المقاطعة لن تخدم بواسطة المعالج لعدم وجود ماسك في مدخل كل خط من هذه الخطوط .

13-4-2 الخط TRAP

خط المقاطعة TRAP يتميز بميزة خاصة عن الخطوط السابقة وهي أن هذا الخط لا يمكن حجبه أو إخفاوه أو تعطيله بأي واحد من الأوامر السابقة وهي SIM أو DI وكذلك لا يحتاج للأمر EI لتنشيطه . لذلك فإن هذا الخط يسمى Nonmaskable interrupt كما أشرنا سابقاً في عمليات المقاطعة الخطيرة مثل الحريق أو إنقطاع التيار الكهربائي أو غير ذلك من العمليات الصناعية الخطيرة . هذا الخط كما هو موضح في شكل (13-2 و 13-3) لا بد وأن يرتفع من صفر إلى واحد ويستمر واحدا high إلى أن يقبله المعالج وإذا نزل إلى الصفر قبل أن يقبله المعالج فلن يكون له أي تأثير .

3-4-3 الخط INTR

آخر خط من خطوط المقاطعة في المعالج 8085 هو الخط INTR . هذا الخط يعتبر من خطوط المقاطعة التي يمكن إخفاوها أو وضع قناع عليها أى أنها عن طريق الأمر DI ولا تقبل أو يتم خدمتها إلا إذا كان علم المقاطعة maskable IF فعالاً عن طريق الأمر EI كما في شكل (3-13) . سنحاول فهم ما يقوم به المعالج وما يحتاج إليه من دوائر خارجية من خلال تتبعنا للخطوات التي يقوم بها المعالج عندما يشعر بأن الخط INTR واحد high وهذه الخطوات كما يلى :

الخطوة الأولى : في أثناء تنفيذ كل أمر من أوامر أي برنامج يقوم المعالج باختبار الخط INTR هل يساوى واحداً أم صبراً ، وذلك من تقاء نفسه وتبعاً لتركيبيه المنطقى .

الخطوة الثانية : إذا كان هذا الخط صبراً فإن المعالج يذهب لتنفيذ الأمر التالي وأما إذا كان واحداً high فإن ذلك يعني طلباً للمقاطعة . عندما إذا كان علم المقاطعة يساوى واحداً عن طريق الأمر EI فإن المعالج سيقبل المقاطعة ويتمكن من تنفيذ الأمر الحالى ويدفع بمحطويات عداد البرنامج إلى المكادسة stack حتى يتمكن من العودة إلى نفس المكان الذي خرج منه .

الخطوة الثالثة : يقوم المعالج بجعل علم المقاطعة IF صبراً لمنع أي مقاطعة من المصادر الأخرى (بالطبع ما عدا المقاطعة من الخط TRAP) ثم يجعل الخط INTA فعالاً يجعله يساوى صبراً low . الخط INTA عندما يكون فعالاً يعني أن المعالج قد قبل المقاطعة Interrupt Acknowledge وهو في انتظار تحديد المكان الذي سيقفز إليه ليبدأ تنفيذ برنامج خدمة المقاطعة وذلك لأنه وكما هو مبين في شكل (3-9) فإن هذا الخط ملحق به ثمانية عناوين يمكن القفز إلى أي واحد منها يتم تحديده للمعالج عند طلب المقاطعة . هذه العناوين مرقمة من صفر إلى سبعة ويتم القفز إلى أي واحد منها عن طريق تنفيذ الأمر RST n حيث n هي رقم العنوان . هذا الأمر يعطى للمعالج عن طريق دائرة خارجية سنعرفها بعد قليل حيث تستخدم الإشارة low على الخط INTA لتنشيط هذه الدائرة لتعطى المعالج الأمر RST n ورقم العنوان المطلوب القفز إليه .

الخطوة الرابعة : عندما يقرأ المعالج الأمر RST n من على مسار العناوين فإنه يقفز إلى العنوان المحدد بهذا الأمر ليبدأ تنفيذ برنامج خدمة المقاطعة من هناك .

الخطوة الخامسة : قبل الانتهاء من برنامج خدمة المقاطعة يجب أن نضع الأمر EI ثم الأمر RET في نهايته حتى تنشط المقاطعة فيصبح المعالج جاهزاً لاستقبال المقاطعات الأخرى عند عودته إلى مكانه الذي خرج منه في البرنامج الأصلي عن طريق إسترداد محتويات عداد البرنامج التي دفعها إلى المكادسة .

4-4-4 كيف يتم تحديد العنوان الذي سيتم القفز إليه في حالة المقاطعة ؟ INTR

كما ذكرنا في الخطوة الرابعة فإن المعالج بعد أن يجعل الخط INTA فعالاً يكون في إنتظار قراءة أمر معين من على مسار البيانات وهذا الأمر هو الأمر RST n الذي يتكون من بait واحد تحتوى شفرة لرقم العنوان الذي سيتم القفز إليه . شكل (13-9) يبين جدول لجميع حالات الأمر RST n والعنوان المصاحب لكل حالة حيث نلاحظ من هذا الجدول أن البتات D3 و D4 و D5 تمثل الرقم n في كل حالة . شكل (13-10) يبين دائرة مقرحة لإدخال شفرة الأمر n على RST n من مسار البيانات فور نزول الخط INTA إلى الصفر . تتكون هذه الدائرة أساساً من أى شريحة بوابات ثلاثة المنطق ولتكن الشريحة 74244 مثلاً والتي تحتوى على ثمانى بوابات من هذا النوع . يوصل خط تنشيط هذه البوابات بالخط INTA القادم من المعالج ويوصل خرج الشريحة على مسار البيانات . يوصل جميع خطوط الدخل بالواحد high ما عدا الثلاثة خطوط D5, D4, D3 فتوصل على مفاتيح تضبط بحيث تعطى الشفرة المناسبة للرقم n المطلوب مع الأمر RST n كما في الجدول المبين في شكل (13-9) . شكل (13-10) يبين هذه الدائرة وقد تم ضبط هذه المفاتيح الثلاثة لتعطى الأمر 5 RST .

الأمر RST n	D7	D6	D5	D4	D3	D2	D1	D0	شفرة الأمر	العنوان
RST 0	1	1	0	0	0	1	1	1	C7	0000
RST 1	1	1	0	0	1	1	1	1	CF	0008
RST 2	1	1	0	1	0	1	1	1	D7	0010
RST 3	1	1	0	1	1	1	1	1	DF	0018
RST 4	1	1	1	0	0	1	1	1	E7	0020
RST 5	1	1	1	0	1	1	1	1	EF	0028
RST 6	1	1	1	1	0	1	1	1	F7	0030
RST 7	1	1	1	1	1	1	1	1	FF	0038

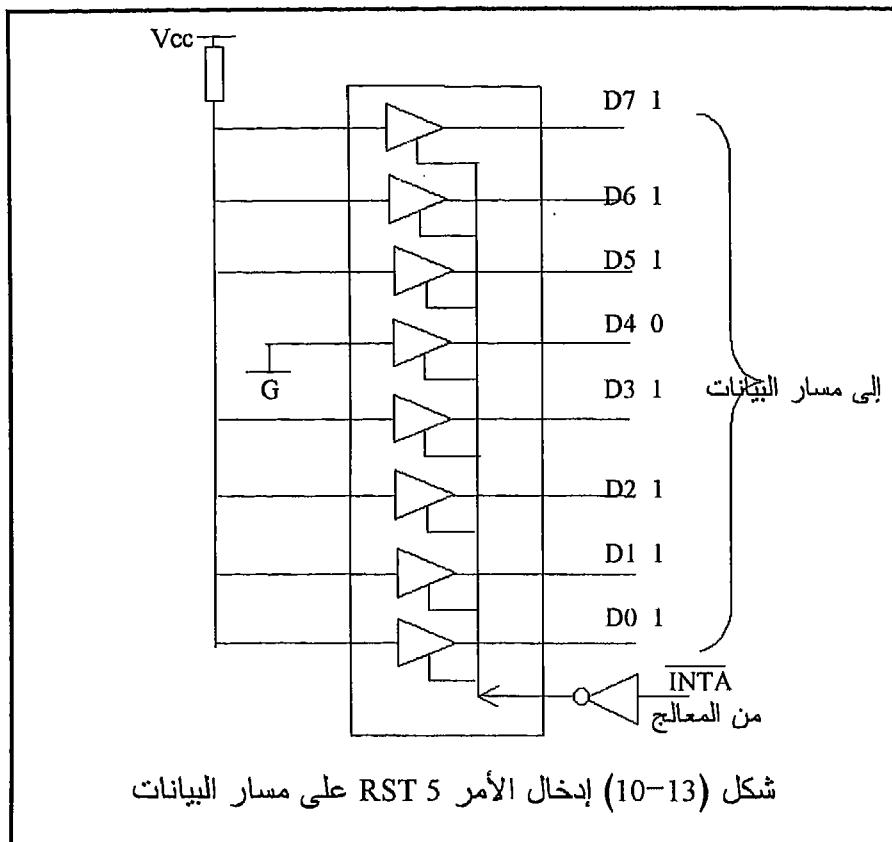
شكل (13-9) الأمر RST n شفرته وعنوان القفز

لاحظ أن الدائرة الموجودة في شكل (13-10) تكافئ تماماً بوابة إدخال تم الاستغناء عن عمليات تشفير عنوان لها واستخدم الخط INTA كخط فعالية لهذه البوابة بحيث أن هذه البوابة ستضع دخلها على مسار البيانات فقط عندما يكون الخط INTA فعالاً .

إن الأمر RST n يمكن تنفيذه من خلال البرنامج وليس بالضرورة أن ينفذ عن طريق إدخاله على مسار البيانات كما رأينا ، فإنه يمكن في أي مكان في البرنامج

أن تنفذ الأمر n RST حيث سيقفز المعالج إلى العنوان المتوافق مع الرقم n وسينفذ برنامج المقاطعه الموجود هناك تماما كما لو كان المعالج تمت مقاطعته من الخارج من على الخط INTR وهذه تسمى مقاطعة عن طريق البرمجة software interrupt . بذلك تكون قد إنتهينا من الطرق المختلفة لمقاطعة المعالج

. 8085



شكل (13-10) إدخال الأمر 5 RST على مسار البيانات

Z80 5-13 مقاطعة المعالج

1-5-13 الخط NMI

هناك خطان أساسيان لمقاطعة المعالج Z80 وهما الخط NMI على الطرف رقم 17 في الشريحة والخط INT على الطرف رقم 16 . بالنسبة للخط NMI والذى يعني مقاطعة غير محجوبة Nonmaskable Interrupt فإنه فعال عندما يكون صفرًا low ولا يهم أن يبقى صفرًا لكي يجيب المعالج على المقاطعة ولكن يكفى

أن يهبط الجهد على هذا الطرف من الوارد إلى الصفر لكي يسجل المعالج طلب المقاطعة ، أى أن الحافة الهابطة هي الحافة المهمة لهذا الطرف . عندما ينتهي المعالج من تنفيذ أى أمر فإنه يختبر خط المقاطعة NMI فإذا كان فعالا يقوم فورا بعمل الآتى :

1. يقوم المعالج بدفع محتويات عداد البرنامج في المكدة حتى يتمكن من العودة إلى نفس المكان الذى خرج منه فى البرنامج الأساسى قبل طلب المقاطعة مثل subroutines.

2. القلاب IFF1 يعتبر علم المقاطعة فى المعالج Z80 بحيث لا يسمح بالمقاطعة إلا إذا كان هذا القلاب يساوى واحد . فى البرنامج الأساسى عادة يقوم المبرمج بوضع هذا القلاب إما واحد أو صفرًا على حسب ظروف البرنامج إذا كان سيسمح بالمقاطعة من على الخط INT أم لا ، ولذلك فإن المعالج يقوم بتخزين قيمة هذا القلاب فى قلاب آخر IFF2 حتى إنه عندما يرجع إلى البرنامج الأساسى بعد خدمة المقاطعة يسترجع القيمة الأصلية للقلاب IFF1 من القلاب IFF2 والتى كانت قائمة قبل المقاطعة . يقوم المعالج بهذه العملية نتيجة لأنه سيغير من قيمة القلاب IFF1 كما سنرى .

3. يوضع القلاب IFF1 يساوى صفرًا وبذلك تمنع أو تحجب أى مقاطعة من على الطرف INT وذلك بالطبع نتيجة لأهمية المقاطعة NMI وأولويته على المقاطعة INT ولذلك فإنه لا يسمح بالمقاطعة من على هذا الطرف إلا فى حالات الطوارئ كما ذكرنا سابقا ولذلك فإنه بمجرد أن يبدأ المعالج فى خدمة هذه المقاطعة فإنه تمنع أى مقاطعة أخرى من أى طرف آخر .

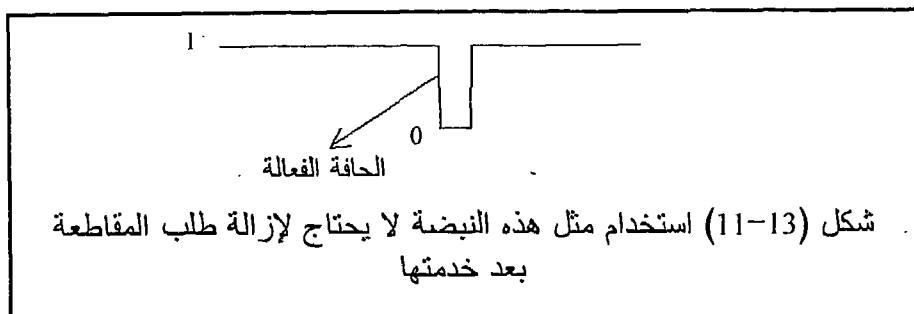
4. يقفز المعالج إلى العنوان H0066 فى الذاكرة والذى من المفروض أن يبدأ من عنده برنامج خدمة المقاطعة للطرف NMI .

5. لابد وأن ينتهى برنامج خدمة المقاطعة بالأمر RTI والذى يعني عودة من مقاطعة Return from Interrupt والذى على أثره يقوم المعالج باسترجاع محتويات عداد البرنامج من المكدة فيرجع إلى البرنامج الأساسى ولنفس المكان الذى تمت عنده المقاطعة ، كما يقوم باسترجاع محتويات القلاب IFF1 من القلاب IFF2 بذلك يعود كل شيء إلى حالته قبل المقاطعة تقريبا .

6. إذا كان برنامج خدمة المقاطعة سيغير من قيمة أى من المسجلات المهمة فى البرنامج الأساسى فإن مسؤولية دفع قيم هذه المسجلات إلى المكدة فى بداية برنامج خدمة المقاطعة ثم استرجاعها مرة أخرى فى نهايتها تقع على المبرمج وذلك لأن المعالج لا يقوم بتخزين كافة المسجلات .

7 - إذا بقى الخط NMI فعالا حتى عند الانتهاء من الخدمة والعودة إلى البرنامج الأساسى فإن ذلك سيسبب مقاطعة أخرى وهذه بالطبع حالة غير مرغوب فيها ، لذلك فإن إزالة طلب المقاطعة من على الخط NMI بعد الانتهاء من برنامج

الخدمة تقع أيضا على عاتق المستخدم وعليه أن يأخذها في اعتباره . أيسر الطرق لذلك هي استخدام نبضة - واحد صفر واحد - كالمبينة في شكل (13-11) لطلب المقاطعة وذلك لأنه كما ذكرنا فإن الخط حساس للحافة الهابطة ولا فائدة من إبقاء الخط أو مسكه على حالة الصفر ، لذلك فإن استخدام مثل هذه النبضة ستغنى المستخدم عن إضافة دوائر أخرى تزيل طلب المقاطعة إذا كان سيبقى ممسوكا على الصفر .



13-5-2 الخط INT

خط المقاطعة INT يدخل على الطرف رقم 16 في المعالج Z80 . المقاطعة على هذا الخط يمكن حجبها باستخدام الأمر DI أى Disable Interrupt والذى يجعل علم المقاطعة IFF1 يساوى صفرًا ، بذلك لا يمكن قبول أى مقاطعة على هذا الخط . يمكن تنشيط المقاطعة مرة ثانية باستخدام الأمر EI والذى يعني Enable Interrupt حيث يجعل علم المقاطعة IFF1 يساوى واحدا وبذلك فإن أى مقاطعة على الخط INT تقبل ويقوم المعالج بخدمتها . الخط INT يكون فعالا عندما يكون صفرًا low ولا بد لكي تقبل المقاطعة من على هذا الخط أن يظل صفرًا إلى أن تقبل المقاطعة ، أى أن هذا الخط ينشط أو يكون فعالا بالمستوى صفر وليس بالحافة كما رأينا في حالة المقاطعة NMI . هناك ثلاثة طرق للتعامل مع الخط INT وهى الطريقة 0 والطريقة 1 والطريقة 2 ويمكن تحديد أو اختيار أى من الطرق الثلاثة باستخدام واحد من الأوامر التالية ، IM1, IM0, IM2، وذلك من داخل البرنامج الأساسي . لاحظ أن المعالج Z80 عند بداية تشغيله أو إعادة وضعة أى RESET يكون في الطريقة 0 .

الطريقة 0 للتعامل مع الخط INT : هذه الطريقة تمثل تماما طريقة مقاطعة المعالج 8085 من على الخط INTR والتي سبق شرحها بالتفصيل حيث أنه عندما يستقبل المعالج Z80 طلب مقاطعة على الخط INT (طرف 16) فإنه يقفز إلى واحد من ثماني عناوين في الذاكرة يقوم المستخدم بتحديده للمعالج عند طلب

المقاطعة . هذه العناوين الثمانية سبق تحديدها في شكل (13-9) ويمكن مراجعتها للتذكرة (لذلك ننصح بقراءة الجزء 13-4 والجزء 13-4-4) . عندما يستقبل المعالج Z80 طلب مقاطعة على الخط \overline{INT} وعندما يكون في الطريقة 0 أى أنه سبق تنفيذ الأمر IM0 فإن المعالج يقوم بتنفيذ الخطوات التالية:

1. يقوم المعالج بتصفير علم المقاطعة IFF1 لحجب أى مقاطعات أخرى من أى أجهزة أخرى تطلب على نفس الخط \overline{INT} .

2. يقوم المعالج بجعل الخطين \overline{IORQ} و $\overline{M1}$ في حالة فعالية ، أى أن كل واحد من هذين الخطين يصبح صفراء low ، والحالة الوحيدة التي يصبح فيها هذان الخطان صفراء معاً وفي نفس الوقت هي حالة المقاطعة على الخط \overline{INT} بالطريقة 0 والتي نحن بصددتها الآن . لاحظ أن الخط \overline{IORQ} معناه طلب قراءة من جهاز إدخال ويقوم المعالج بتنشيط هذا الخط لأنه سيكون في هذه الحالة في إنتظار قراءة الأمر n RST كما شرحنا في حالة المعالج 8085 . وأما الخط $\overline{M1}$ فإنه يكون فعالاً فقط عند قراءة شفرة أى أمر ، وحيث أن المعالج في حالتنا هذه بالذات يقرأ شفرة الأمر n RST من على بوابة إدخال فإنه سيكون هو أيضاً فعالاً في نفس اللحظة التي سيكون فيها الخط \overline{IORQ} فعالاً .

3. على المستخدم أن يستغل ظاهرة أن الخطين $\overline{M1}$ و \overline{IORQ} يكونان صفراء معاً في هذه الحالة فقط لتشغيل بوابة إدخال يدخل عليها شفرة الأمر n RST والذى يحدد للمعالج أى واحد من عناوين القفز الثمانية سيقفز إليه . هذه البوابة موضحة في شكل (13-12) . لاحظ أن الفرق الوحيد بين المعالج Z80 والمعالج 8085 هو فقط في طريقة تشغيل هذه البوابة حيث كما رأينا في حالة المعالج 8085 يستخدم الخط \overline{INTA} لتشغيل هذه البوابة كما في شكل (13-10) .

4. يقرأ المعالج بوابة الإدخال ويفك شفرة الأمر n RST ليعرف العنوان الذي سيتم القفز إليه .

5. يقوم المعالج بدفع محتويات عدد البرنامج في المكدة .

6. يقفز المعالج إلى برنامج خدمة المقاطعة بناء على العنوان الذي قرأه في الخطوة 4 .

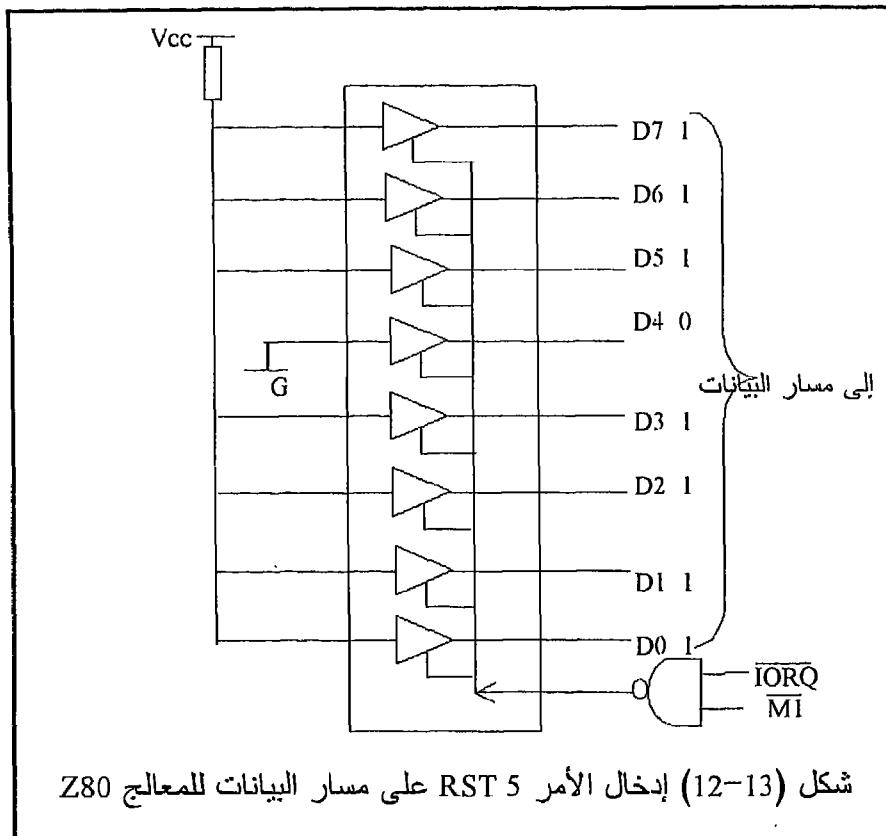
7. بعد الانتهاء من تنفيذ برنامج خدمة المقاطعة يرجع المعالج إلى نفس المكان في البرنامج الأصلي عن طريق جلب محتويات عدد البرنامج مرة ثانية من المكدة .

الطريقة 1 للتعامل مع الخط \overline{INT} : هذه الطريقة بسيطة جداً حيث أنها تشبه تماماً طريقة المقاطعة من على الخط \overline{NMI} حيث أن هناك عنواناً واحداً فقط وهو العنوان H 0038H يتم القفز إليه في حالة المقاطعة بهذه الطريقة . لاحظ أنه لابد من تنفيذ الأمر IM1 قبل طلب المقاطعة في البرنامج الأساسي ، كما أن علم المقاطعة لابد وأن يكون واحداً ، أى أنه تم تنفيذ الأمر EI أيضاً لتنشيط هذا العلم وحتى

يمكن قبول المقاطعة . عند المقاطعة بهذه الطريقة أيضا يقوم المعالج بدفع محتويات عداد البرنامج إلى المكذبة وعلى المبرمج دفع باقي المسجلات إن أراد في بداية برنامج خدمة المقاطعة حيث أن المعالج لا يقوم بذلك .

الطريقة 2 للتعامل مع الخط INT : يدخل المعالج في هذه الطريقة بناء على تنفيذه للأمر IM2 ، وعند استقباله لطلب مقاطعة يقوم بالخطوات التالية :

1. يقوم المعالج بوضع صفر في علم المقاطعة IFF1 لمنع أي مقاطعة أخرى مثل الطرق السابقة .



شكل (13-12) إدخال الأمر 5 RST على مسار البيانات للمعالج Z80

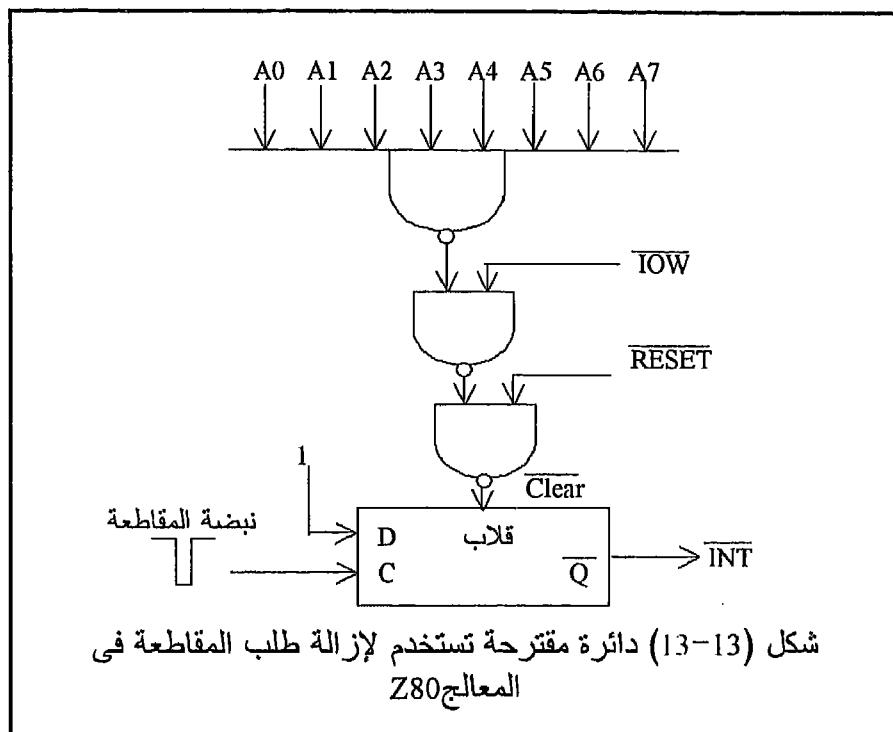
2. مثل الطريقة 0 يقوم المعالج بجعل الخطين $\overline{M1}$ و \overline{IORQ} فعالين بجعل كل منها يساوى صبرا . على المبرمج مثل ما شرحنا فى حالة الطريقة 0 أن يستعمل ذلك لإدخال شفرة معينة إلى مسار البيانات من على بوابة إدخال كالموضحة فى شكل (12-13) .

3. يقوم المعالج بتكوين عنوان من 16 بتا وهذا العنوان يتكون من جزأين : الجزء الأول من العنوان وهو A0 إلى A7 هو الشفرة التى تمت قراءتها من على

بوابة الإدخال كما في الخطوة 2 وأما الجزء الثاني من العنوان A8 إلى A15 فهو محتويات المسجل I الموجود في المعالج لهذا الغرض والذي يتكون من 8 بิตات. عنوان برنامج خدمة المقاطعة في هذه الحالة يكون موجوداً في البايت المحددة بالعنوان السابق والبايت التي تليها .

4. بعد معرفة هذا العنوان يقفز إليه المعالج لينفذ برنامج خدمة المقاطعة الموجود هناك وذلك بالطبع بعد دفع محتويات عداد البرنامج إلى المكادسة حتى يستطيع العودة إلى نفس المكان في البرنامج الأصلي بعد الانتهاء من برنامج خدمة المقاطعة .

5. يمكن أن نوضح ذلك بالمثال التالي : افترض أن محتويات المسجل I هي 10H ، وعند قراءة بوابة الإدخال المقصودة وجد بها الرقم 45H . لذلك سيكون المعالج العنوان التالي 1045H ويدرك إلى هذه البايت ليقرأ محتوياتها ولتكن مثلاً 00H ثم يذهب إلى البايت التالية لها أي البايت 1046H فيقرأ محتوياتها ولتكن مثلاً E1H . من محتويات هاتين البايتين يتكون لدى المعالج عنوان برنامج خدمة المقاطعة الذي يكون E100H في هذا المثال .



شكل (13-13) دائرة مقترحة لازالة طلب المقاطعة في المعالج Z80

كما رأينا فإنه لابد وأن يبقى الخط INT ممسوكا على الصفر حتى تتم خدمته بواسطة المعالج وإلا فإنه إن رجع إلى الواحد قبل أن يتعرف المعالج عليه فلن تتم خدمته وذلك لأن الخط INT من النوع الذي يكون فعالا على المستوى صفر وليس على أي حافة . لذلك فإنه يستحسن إدخال طلب المقاطعة على الطرف INT عن طريق قلاب كما هو مبين في شكل (13-13) . وجود مثل هذا القلاب في مدخل المقاطعة INT ومسكه على الصفر عند طلب المقاطعة يستلزم أن يقوم المستخدم بإزالة هذا الصفر (طلب المقاطعة) عند نهاية برنامج الخدمة . شكل (13-13) يبين أيضا دائرة مقرحة لهذا الشأن حيث يتم إخراج أي معلومة على بوابة الإخراج FFH مثلًا التي نتيجة لتشفيتها تعطى واحد على الخط INT أي تزيل المقاطعة . أي أنه بمجرد تنفيذ الأمر OUT FFH في نهاية برنامج خدمة المقاطعة سيزال طلب المقاطعة من على الطرف INT .

6- تمارين

1. ما هو المقصود بطرق الأبواب لخدمة الأجهزة ؟
2. ما هو الفرق بين طريقة طرق الأبواب لخدمة الأجهزة والمقاطعة ؟
3. أشرح بعض التطبيقات التي تستخدم فيها المقاطعة ؟
4. ماذا يفعل أي معالج عادة حال إستقباله لأمر مقاطعة ؟
5. كم عدد خطوط المقاطعة لدى المعالج 8085 التي يمكن مقاطعته من عليها ؟
6. كم عدد خطوط المقاطعة لدى المعالج Z80 التي يمكن مقاطعته من عليها ؟
7. الخط TRAP فى المعالج 8085 ، ماذا يكافئه فى المعالج Z80 ؟
8. الأوامر DI و EI للمعالج 8085 يستخدمان عادة فى برامج خدمة المقاطعة ، ما هو الغرض من استخدامهما ؟
9. أشرح دور الأمرين SIM و RIM فى المقاطعة على الخطوط RST7.5, RST6.5, RST5.5, Z80 و المعالج 8085 ؟
10. ما هو الفرق بين المقاطعة على الخط INTR والخطوط الأخرى فى المعالج Z80 و المعالج 8085 ؟
11. أشرح كيفية تحديد مكان برنامج خدمة المقاطعة للمعالج 8085 فى حالة المقاطعة على الخط INTR ؟
12. أشرح كيفية إدخال شفرة الأمر RST n إلى المعالج 8085 و Z80 ؟
13. من أوامر الشريحة Z80 الأوامر ، IM0 و IM1 و IM2 ، ماذا تعنى هذه الأوامر وهل لها نظير فى المعالج 8085 ؟ قارن بين الطرق الثلاثة لمقاطعة المعالج Z80 على الخط INT وما يناظرها فى المعالج 8085 ؟
14. قارن بين الشكلين (13-10) و (13-12) ؟

15. مطلوب استخدام المعالج للتحكم فى نظام درجة حرارة غرفة فى أحد المصانع وإظهار هذه الدرجة على شريحتين ذات 7 قطع , 7 segment . ارسم هذا النظام وإكتب برنامجه مرة مستخدما نظام طرق الأبواب ومرة باستخدام المقاطعة ؟

الفصل الرابع عشر

التركيب الميكانيكي للمعالج

Intel 8086/8088

1-14 مقدمة

نقدم في هذا الفصل شرحا تفصيلا للتركيب الهيكلي للمعالج intel8086 وزميله intel8088 وذلك من الداخل حيث سندرس محتوياتهما من المسجلات والعدادات ومن الخارج حيث سنلقي نظرة سريعة على وظيفة كل طرف من أطرافهما . إن هذين المعالجين يعتبران أهم المعالجات ذات ال 16 بت وذلك لاستخدامهما في الحاسوب IBM الذي كان من أول الحاسوبات الشخصية التي ظهرت في السوق ثم سادت وفرضت نفسها على كل المستخدمين للحواسيب . من ضمن هذا الجيل من المعالجات ظهر أيضا المعالج MC68000 والمعالج Z8000 ولكن كما ذكرنا كان أكثرها شيوعا وأوفرها حظا في الاستخدام هو المعالج intel8086 والذي نحن بصدده دراسته هنا .

المعالجين 8086/8088 كل منهما كما رأينا يتعامل مع بيانات مقدارها 16 بت ، ولكن هناك نقطة خلاف أساسية ووحيدة بين هذين المعالجين وهي كيفية التعامل مع هذه البيانات . إن المعالج 8086 يتعامل مع البيانات في داخله وفي خارجه على أساس أنها 16 بت ، أي أنه له مسار بيانات خارجي مقداره 16 بت يستطيع من خلاله التعامل مع الأجهزة الخارجية مثل الشاشة والذاكرة ولوحة المفاتيح على أساس 16 بت ، فيرسل مثلا معلومة من 16 بت إلى الشاشة في مرة واحدة ويستقبل معلومة من 16 بت من أي بوابة إدخال . هذا المعالج ، 8086 يتعامل داخليا أيضا بنفس الطريقة حيث تنتقل البيانات داخليا بين جميع المسجلات بعضها البعض أو مع وحدة الحساب والمنطق على مسار بيانات مقداره 16 بت أيضا . أما المعالج 8088 فإنه تماما مثل نظيره 8086 في التعاملات الداخلية حيث ينقل البيانات داخله على مسار بيانات مقداره 16 بت ، بينما يختلف عن نظيره 8086 في التعاملات الخارجية حيث أن مسار البيانات الخارجي له يتكون من 8 بت فقط ، لذلك فإنه يرسل أو يستقبل بيانات 8 بت فقط مع الأجهزة الخارجية ، وهذه كما ذكرنا هي نقطة الخلاف الوحيدة بين المعالجين كما سنرى فيما بعد . ونعتقد أن الرقم 6 في المعالج 8086 يذكرنا بأنه يتعامل دائما من خلال مسار بيانات 16 بت ، بينما الرقم 8 الأخير في المعالج 8088 فإنه يذكرنا بأن التعامل يكون من خلال مسار بيانات 8 بت فقط مع الأجهزة الخارجية . ويخطر ببالنا سؤال مهم هنا وهو: ما هو الداعي للمعالج 8088 إذا كان نظيره 8086 قد قام بالمهام بكفاءة أحسن وبالتأكيد أسهل حيث يتعامل خارجيا وداخليا من خلال مسار بيانات 16 بت ؟ والإجابة على ذلك هي أن المعالجات 16 بت ظهرت في منتصف الثمانينيات وحلت محل المعالجات 8 بت في جميع الحاسوبات وجميع التطبيقات وبعد أن كانت المعالجات 8 بت قد انتشرت في السوق واستخدمت في كثير من أجهزة الحاسوبات وفي الكثير من التطبيقات أيضا مما

تسبب في أن كل هذه الأجهزة القديمة (8 بت) أصبحت عديمة الفائدة بعد أن فكر كل المستهلكين في الأخذ بالمعالجات الجديدة (16 بت). لذلك كان التفكير في معالج وسيط يقلل من حجم الخسارة في عملية الانتقال إلى المعالج 8086 ، فكان الحل هو معالج يتعامل داخلياً على أساس 16 بت للاستفادة بميزات الـ 16 بت ويعامل خارجياً على أساس 8 بت لتسهيل عملية المواجهة مع الأجهزة الخارجية الموجودة أصلاً في الحاسوب والتطبيقات التي كانت تتعامل مع المعالجات 8 بت وبأقل تكلفة ممكنة ، فكان ذلك المعالج الوسيط هو المعالج 8088 ، ولقد نزلت في هذا الوقت أيضاً الكثير من البرامج التي تقوم بتحويل برامجيات المعالجات 8 بت إلى صورة تناسب المعالجات 16 بت كمرحلة انتقال و حتى لا يعاد صياغة هذه البرمجيات من جديد .

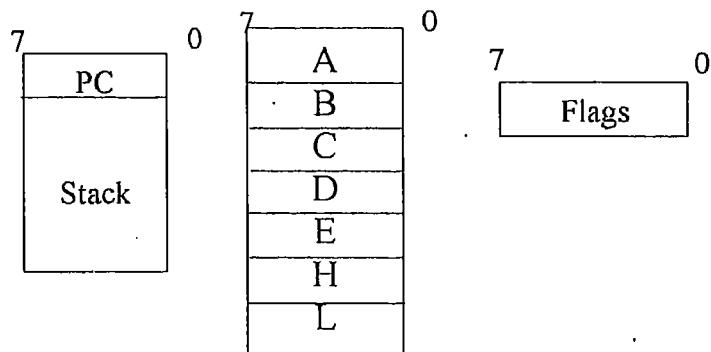
14-2 نظرة داخلية على محتويات المعالجين 8086/8088

كما علمنا من قبل وعند دراستنا لشريحة المعالجات 8 بت أن أي معالج في النهاية يمكن النظر إليه على أنه مجموعة من المسجلات والعدادات بجانب وحدة الحساب والمنطق حيث أنها أهم المكونات الداخلية في المعالج ، أما فكرة عمل المعالج ، أي معالج ، فهي (كما سبق وشرحناها أيضاً) إحضار شفرات الأوامر من الذاكرة وتفيذها بنفس التتابع المسجلة به في البرنامج . أي أن الفكرة ثابتة ولكن التطور يكون دائماً في المكونات حيث تتغير المسجلات ووحدة الحساب والمنطق من 8 بت إلى 16 بت إلى 32 بت إلى 64 بت وتتغير تكنولوجيا التصنيع نفسها مع الزمن فترتاد السرعة بدرجة كبيرة ولكن فكرة العمل تظل كما هي ثابتة . شكل (14-1) يبين تطور المسجلات في المعالجات 8008 و 8085 و 8086 . نلاحظ من هذا الشكل أن عدد المسجلات كان 6 مسجلات كل منها 8 بت بخلاف المركم في المعالج 8008 وأما المعالج 8085 فيحتوى نفس العدد من المسجلات ولكن الجديد هو أن هذه المسجلات يمكن فى بعض العمليات ازدواجها واستخدامها كمسجلات 16 بت كما رأينا سابقاً ولكن المركم كما هو 8 بت ، وأما في المعالج 8086 فإنه يحتوى أيضاً نفس العدد من المسجلات العامة والتى اختلفت أسماؤها قليلاً لتناسب استخدامها كمسجلات 16 بت أو 8 بت ، فمثلاً المسجل B أصبح اسمه BX في حالة استخدامه كمسجل 16 بت أو BL في حالة استخدام النصف الأول منه كمسجل 8 بت و BH في حالة استخدام النصف الأعلى منه كمسجل 8 بت ، نفس الكلام مطبق على باقى المسجلات العامة وهى المسجلات C, D . الجديد أيضاً أن المركم مطبق عليه نفس الكلام السابق فيمكن استخدامه كمسجل 16 بت (AX) أو مسجلين كل منهم 8 بت (AL, AH) . نلاحظ أيضاً أن المكادسة stack كانت موجودة بداخل المعالج 8008 ثم انتقلت لتصبح

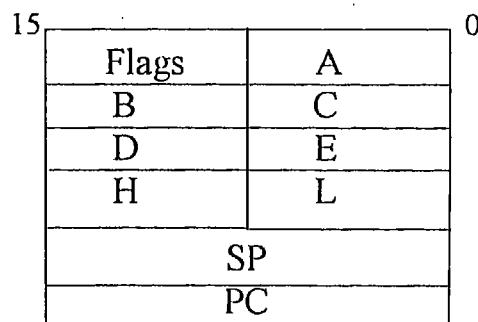
جزءاً من الذاكرة يشار إلى قمتها أو أول مكان فاضي فيها بمحطيات المسجل SP أو مؤشر المكدة وذلك في المعالجات 8085 و 8086 . أما عدد البرنامج PC فكان 8 بت في المعالج 8008 وأصبح 16 بتاً في المعالجات 8085 و 8086 وأصبح اسمه مؤشر الأوامر IP في المعالج 8086 وهناك فرق كبير بين الاسم "عدد البرنامج" و "مؤشر الأوامر" بالرغم من التمايز في الوظيفة ولكن في حالة المعالج 8085 فإنه يتعامل مع ذاكرة مقدارها 64 كيلو بايت وأما في حالة المعالج 8086 فإنه يتعامل مع ذاكرة مقدارها 1 ميجابايت من خلال فكرة زكية وهي فكرة تجزيء الذاكرة التي سنشرحها بعد قليل إن شاء الله . نلاحظ أيضاً من شكل (14-1) أن المعالج 8086 يحتوى على مسجلات أخرى لم تكن موجودة في سابقيه وهي المسجلات BP, SI, DI, CS, DS, SS, ES وكلها مسجلات 16 بت سنتعرف على وظيفتها كل منها بعد قليل . نلاحظ أيضاً أن مسجل الأعلام أصبح 16 بت أيضاً بدلاً من ثمانية مما يتبين بأنه سيكون هناك الكثير من الأعلام وبالتالي مقدرة أكثر على البرمجة وعدد أكثر من الأوامر . من الملاحظات المهمة أيضاً في شكل (14-1) هي احتواء المعالج 8008 على المكدة كمجموعة من المسجلات موجودة بداخل المعالج نفسه في حين أصبحت هذه المكدة جزء من الذاكرة في المعالجات التي تلت ذلك مما أمكن معه تكبير المكدة لأى كمية مطلوبة .

3-14 نظرية تفصيلية على مسجلات المعالج 8086/8088

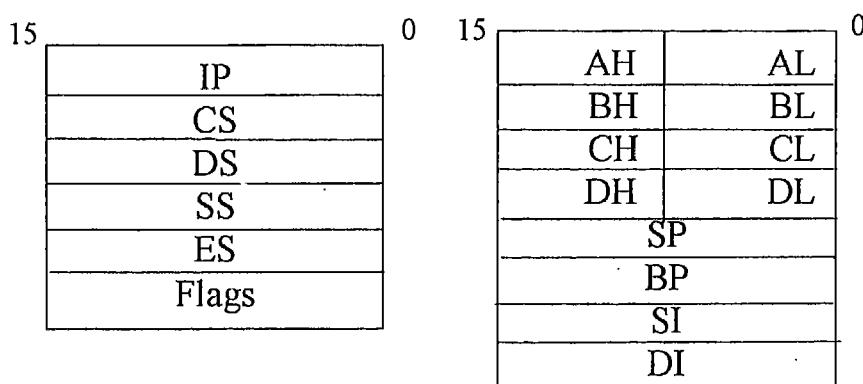
لقد استخدم مصممو المعالجين 8086/8088 فكرة زكية كان لها أكبر الأثر في زيادة سرعة وكفاءة هذين المعالجين وهذه الفكرة هي انقسام هذه المعالجات إلى وحدتين أساسيتين لكل منها وظيفة مختلفة تماماً عن الوحدة الأخرى .
الوحدة الأولى هي وحدة التنفيذ (EU) Execution Unit وهي خاصة فقط بتنفيذ الأوامر ولا تتعداها لأى وظيفة أخرى ، والوحدة الثانية هي وحدة مواجهة المسارات (BIU) Bus Interface Unit وهذه أيضاً لها وظيفة محددة وهي جلب الأوامر من الذاكرة ووضعها في طابور أو قائمة انتظار queue في انتظار التنفيذ عن طريق وحدة التنفيذ . هذا التقسيم في الوظائف بين الوحدتين أتاح لوحدة التنفيذ أن تقوم فقط بتنفيذ الأوامر الموجودة في قائمة الانتظار وفي أثناء اشغال الذاكرة ووضعها في القائمة والعمل على أن تكون القائمة مملوءة دائماً بالأوامر التي في انتظار التنفيذ .



مسجلات المعالج intel80008



مسجلات المعالج intel8085



مسجلات المعالج intel8086

شكل (1-14) تطور محتويات المعالجات 8008 و 8085 و 8086

بذلك تم توفير وقت كبير كانت وحدة التنفيذ تتوقف فيه لحين الذهاب إلى الذاكرة وإحضار الأوامر التي عليها الدور في التنفيذ مما كان له أكبر الأثر في زيادة سرعة هذه المعالجات بدرجة كبيرة . يمكن تمثيل مهمة كل من هاتين الوحدتين بمهمة المدير والسكرتارية حيث تكون السكرتارية هي المواجهة للعالم الخارجي حيث تستقبل هي جميع الطلبات والمشاكل من الجمهور واعدادها في ملف كل على حسب ترتيب قدمه ثم تعرض هذا الملف على المدير الذي يقوم بحل هذه المشاكل في حين تعمل السكرتارية على استقبال الطلبات الأخرى لحين أن ينتهي المدير من تنفيذ ما معه من طلبات . بالطبع فإن ذلك يكون له أكبر الأثر في سرعة تنفيذ الطلبات . عن ما لو كان المدير وحده يقوم باستقبال كمية من الطلبات ثم يجلس لتنفيذها وبعد أن ينتهي من تنفيذ هذه الكمية يقوم لاستقبال كمية أخرى وهكذا .

بعض المسجلات داخل المعالج 8086 تتبع وحدة التنفيذ والبعض الآخر يتبع وحدة المواجهة على حسب وظيفة كل مسجل من هذه المسجلات حيث يجب أن تتوقع أن جميع المسجلات العامة AX, CX, BX, DX وسجل الأعلام والمسجلات SP كلها تتبع وحدة التنفيذ وأما المسجلات CS, DS, SS, DI, SI, BP, مؤشر الأوامر IP فتتبع وحدة المواجهة بعدها لوظيفة كل منها كما سنرى .

على ضوء ما ذكرنا في المقدمة عن الفرق بين المعالجين 8086 و 8088 فإننا يجب أن نتوقع أن وحدة التنفيذ EU في كل من المعالجين ستكون نفسها تماما وأما وحدة المواجهة BIU فستختلف في المعالج 8086 عنها في المعالج 8088 حيث أنها في الأول ستتعامل مع مسار بيانات 16 بت بينما ستتعامل مع مسار بيانات 8 بتات في المعالج الثاني وهذا هو وجہ الاختلاف الأساسي بينهما .

14-3-1 المسجلات عامة الأغراض

يحتوى المعالج 8086 على أربع مسجلات عامة الأغراض كل منها 16 بتا وهى المسجلات DX, CX, BX, AX . كل واحد من هذه المسجلات يمكن التعامل معها على أنها مسجلين كل منهم 8 بتات أو مسجل واحد 16 بت . فى حالة التعامل معها على أنها مسجلات 8 بتات فإن النصف الأدنى أو ذو القيمة الصغرى Low significant half يرمز له دائمًا بالرموز التالية DL, CL, BL ، أما النصف الأعلى أو ذو القيمة العليا High significant half فيرمز له AL . لذلك عند وضع معلومة من 16 بت دائمًا بالرموز التالية DH, CH, BH, AH . مثل الرقم C351H فى المسجل BX فإن النصف الأعلى من المعلومة وهو C3 يوضع فى النصف الأعلى من المسجل وهو BH وأما النصف الأدنى من المعلومة وهو 51 فيوضع فى النصف الأدنى من المسجل وهو BL . فيما يلى سنلقي نظرة سريعة على وظيفة كل مسجل من هذه المسجلات :

المسجل AX

المسجل AX هو المركم accumulator وكما سنرى عند دراستنا للغة الأسsembli للمعالج 8086 فإن المركم لن تكون له نفس الأهمية التي رأيناها عند دراستنا للمعالجات 8 بت ، حيث هنا سنرى أنه يمكن إجراء أي عملية حسابية أو منطقية على أي مسجلين مع بعضهما البعض وليس من الضروري أن يكون المركم واحداً منها ، كما أن نتيجة هذه العملية تكون دائماً في المسجل الأول في الأمر ، فمثلاً الأوامر التالية كلها صحيحة :

ADD AX, BX

حيث سيجمع محتويات المسجل AX مع المسجل BX ويوضع النتيجة في المسجل AX الذي هو المركم .

ADD CX, BX

حيث سيجمع محتويات المسجل CX مع المسجل BX ويوضع النتيجة في المسجل CX ، وهكذا . هذا لا زالت عمليات الإدخال والإخراج باستخدام الأمرين IN و OUT تتم عن طريق المركم كما هو الحال في المعالجات 8 بت ولكن بإمكانيات أكثر وكفاءة أحسن كما سنرى عند الدراسة التفصيلية لهذه الأوامر .

المسجل BX

إن المسجل BX بجانب كونه أحد المسجلات العامة التي تستخدم في كل أغراض البرمجة مثل العمليات الحسابية والمنطقية وعمليات الإزاحة والدوران وغيرها فإن له وظيفة أخرى محددة وخاصة به عند تنفيذ بعض الأوامر مثل الأمر XLAT والذي ينشئ جدولًا في الذاكرة أول عنوان فيه هو الموجود في المسجل BX وت تكون عناصر هذا الجدول ب تخزين محتويات النصف الأدنى من المركم BX في عناوين متتالية في الذاكرة تتكون بجمع محتويات المسجل AL مع محتويات المسجل BX كما سنرى عند شرح أوامر لغة الأسsembli فيما بعد . لذلك فإن المسجل BX يحتوى عنوان البداية أو القاعدة Base للجدول الذي يتكون بالأمر XLAT . كما أن المسجل BX يستخدم في أغراض العنونة غير المباشرة حيث يمكن وضع العنوان المراد التعامل معه في ذاكرة البيانات فيه .

المسجل CX

المسجل CX أيضاً بجانب كونه أحد المسجلات العامة مثل المسجلين BX, AX فإن له أيضاً مهمة محددة خاصة به عند تنفيذ بعض الأوامر . فمثلاً عند تنفيذ الأمر LOOP والذي ينفذ حلقة أو مجموعة من الأوامر عدة مرات فإن عدد المرات المراد تنفيذها لهذه الحلقة يوضع في المسجل CX . أي أنه عدد أو للحلقات عند تنفيذ الأمر Counter .

المسجل DX

هذا المسجل أيضاً بجانب كونه أحد المسجلات العامة فله أيضاً وظيفة محددة عند تنفيذ بعض الأوامر ، فعند تنفيذ أمر ضرب رقمين كل منهما 16 بت فإن النصف الأعلى part most significant من النتيجة يوضع في هذا المسجل (الاحظ أن النتيجة ستكون 32 بت) . كذلك عند تنفيذ بعض أوامر الإدخال والإخراج فإن المسجل DX يوضع به عنوان البوابة المراد الإخراج عليها أو الإدخال منها . أى أن هذا المسجل DX أحد وظائفه الخاصة هي أنه يحتوى جزء من البيانات عند تنفيذ أوامر ضرب أو قسمة رقمين كل منهما 16 بتاً . وعلى ذلك فإن الأربع مسجلات السابقة لها أسماء كما رأينا تطابق الرموز التي أطلقنا عليها والوظيفة الخاصة المنوطة بكل واحد من هذه المسجلات والتي سنعيدها كما يلى :

Accumulator	AX	المركم
Base	BX	القاعدة
Counter	CX	العداد
Data	DX	البيانات

هناك أيضاً مجموعة من المسجلات التي يمكن أن تدخل ضمن مجموعة المسجلات العامة حيث أنها تكون تحت تصرف المبرمج ولكنها لها وظيفة محددة أيضاً في عمليات البرمجة فهي تستخدم إما للإشارة إلى أماكن محددة في الذاكرة أو تستخدم في الفهرسة Index عند التعامل مع الذاكرة بهذه الطريقة . هذه المسجلات هي كالتالى :

مؤشر المكذسة أو المسجل Stack Pointer, SP

المكذسة stack هي جزء مقطوع من الذاكرة يخزن فيه عادة البيانات المهمة قبل القفز من البرنامج الأساسي إلى برنامج فرعى أو برنامج مقاطعة والتي ستكون هناك حاجة إليها عند العودة مرة ثانية إلى البرنامج الأساسي بعد إنتهاء البرنامج الفرعى (انظر فصل البرامج الفرعية) أو الانتهاء من خدمة المقاطعة (انظر فصل المقاطعة) . من أهم هذه البيانات مثلاً محتويات مؤشر الأوامر IP حتى يتتسنى لنا العودة لنفس المكان الذي خرجنا منه في البرنامج الأساسي وكذلك محتويات أي مسجل آخر قد نخاف من ضياعها أو تغيرها عند الخروج من البرنامج الأساسي مثل مسجل الأعلام والمركم . هذه البيانات تخزن في المكذسة بالترتيب ويتم استدعاؤها بنفس الترتيب على أساس أن آخر ما تم تخزينه يكون أول ما يتم استدعاؤه (Last In First Out) LIFO ولكنى نعرف حدود هذه المكذسة فإن مسجل مؤشر المكذسة Stack Pointer (SP) يحتوى عنوان آخر مكان تم التخزين فيه في هذه المكذسة .

مسجل مؤشر القاعدة Base Pointer, BP

أحد المسجلات العامة التي تستخدم لعنونة أو للإشارة على بداية مجموعة بيانات أو طابور array بيانات موجود في المكادسة stack .

مسجل الفهرسة SI, DI

يستخدمان في عملية العنونة غير المباشرة (المفهرسة) indirect في الذاكرة memory segmentation registers كما سنرى عند دراسة طرق العنونة المختلفة addressing .

3-2 المسجلات الخاصة

المسجلات الخاصة الموجودة في المعالج 8086/8088 هي مسجل مؤشر الأوامر Instruction Pointer (IP) وأربع مسجلات خاصة بتجزئ الذاكرة سنطلاق عليها اسم مسجلات التجزيء memory segmentation registers . عدد هذه المسجلات أربعة وهي : ES, SS, DS, CS . لكي نأخذ فكرة عن وظيفة هذه المسجلات ، لا بد أن نعرف أولاً ما هو المقصود بتجزئ الذاكرة ؟ ولماذا يتم تجزئ الذاكرة ؟

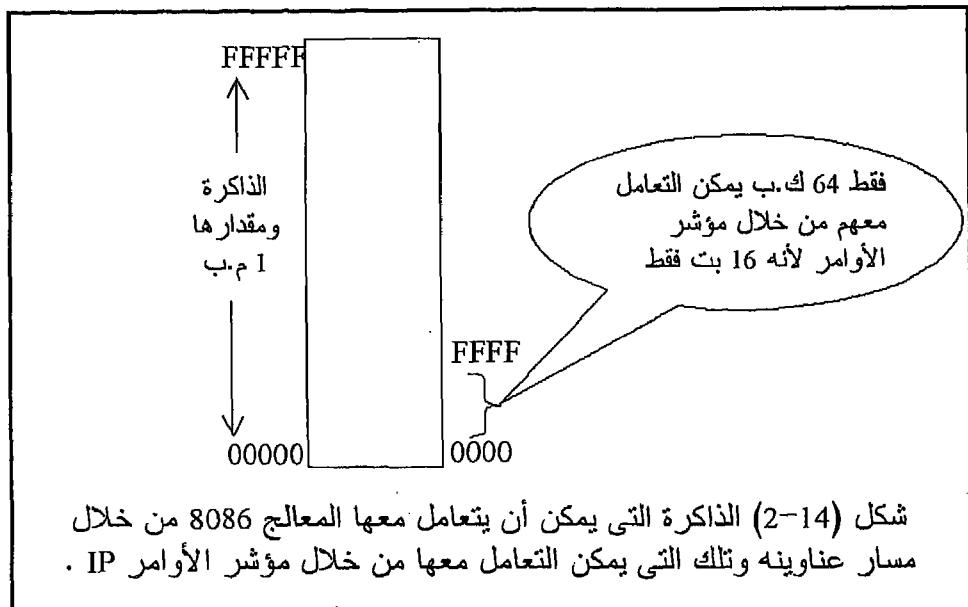
4-4 تجزئ الذاكرة Memory segmentation

مسار العنوانين في المعالج 8086/8088 يحتوى 20 بت أو بمعنى آخر يتكون من 20 خطأ وهذا يعني أن هذا المعالج يستطيع التعامل مع ذاكرة مقدارها 2^{20} أو 1048576 بait أو اختصاراً تكتب 1 ميجابايت (1 M.B) . هذا يعني أنتا تستطيع مثلاً أن تكتب أي برنامج في خلال هذا المدى من الذاكرة والذي يبلغ 1 M.B كما ذكرنا وعلى المعالج أن يحضر أوامر هذا البرنامج من الذاكرة وينفذها بالتتابع بلا أدنى مشاكل . إن هناك مشكلة صعبة تعيق المعالج من إمكانية إحضار الأوامر من الذاكرة بهذه السهولة وذلك لأن عدد البرنامج أو مؤشر الأوامر IP كما أسميه هنا يحتوى على 16 بت فقط ، وكما نعلم أن مهمة المسجل IP هي أنه يحتوى عنوان الأمر الذى عليه الدور فى التنفيذ ، وهذا يعني وبالتالي أن أي أمر يقع في الذاكرة خارج المدى العنوانى صفر إلى 64 ك.B لن يستطيع المعالج إحضاره لأن مؤشر الأوامر IP يتكون من 16 بت فقط مما يعني أن المدى العنوانى الذى يستطيع المعالج التعامل معه من خلال هذا المسجل هو صفر إلى 2^{16} أي 65536 بait في الذاكرة . فما هو الحل لهذه المشكلة ونحن نريد كتابة البرامج في أي مكان في الذاكرة التي يبلغ مداها 1 M.B وليس فقط في أول 64 ك.B؟ شكل (14-2) يبين المدى العنوانى للمعالج 8086 على ضوء

عدد خطوط مسار العنوانين ، والمدى العنوانى الذى يمكن التعامل معه من خلال المسجل IP .

إن حل هذه المشكلة جاء من خلال استخدام فكرة زكية تمكنت كمبرمج من التعامل مع كل المدى العنوانى للذاكرة الذى يبلغ 1M.B بالرغم من استعمال مسجلات 16 بت فقط وكان ذلك من خلال استخدام أربع مسجلات سميت بمسجلات تجزيء الذاكرة memory segmentation registers وكل منها 16 بت ويرمز لها بالرموز التالية ES, SS, DS, CS وهذه الرموز لها دلالات تتطابق مع وظيفة كل مسجل سنعرفها بعد قليل .

كل واحد من هذه المسجلات ، مسجلات التجزيء ، يحتوى عنوان من 16 بت ولكن الظريف هنا أن هذه 16 بت تقابل أعلى 16 بت من مسار العنوانين أى A4 إلى A19 وليس أول 16 بت A0 إلى A15 . أى أن محتويات أى واحد من هذه المسجلات لن تمثل عنواناً حقيقياً في الذاكرة إلا بعد إزاحتها ناحية اليسار بمقدار 4 بت أى بمقدار خانة ستعشرية أو بضربها في الرقم 16 ضرباً عشرياً للحصول على عنوان من 20 بت .



شكل (14-2) الذاكرة التى يمكن أن يتعامل معها المعالج 8086 من خلال مسار عنوانين وتلك التى يمكن التعامل معها من خلال مؤشر الأوامر IP .

فمثلاً بافتراض أن المسجل C S محتوياته كالتالى : CS=0800H فإن العنوان الفعلى المقابل لهذه المحتويات هو 08000H بالإضافة 0H ناحية اليمين أى بإزاحة الرقم 4 ببات ناحية اليسار أو بضربه في الرقم 16 ضرباً عشرياً . وكذلك إذا كانت محتويات المسجل DS كالتالى : DS=12F5H فإن العنوان الفعلى المقابل

لهذه المحتويات هو 12F50H في الذاكرة . هنا يظهر سؤال مهم وهو كيف يتسم إحضار الأوامر من الذاكرة باستخدام مؤشر الأوامر IP الذى يتكون هو الآخر من 16 بت فقط ؟ وهل هذا المسجل له علاقة بمسجلات التجزء ؟

بفرض أن محتويات مسجل التجزء CS هى CS=12F0H ، وأن محتويات مؤشر الأوامر IP=001BH هى IP فما هو العنوان الحقيقى فى الذاكرة للأمر الذى عليه الدور فى التنفيذ ؟ يتعدد هذا العنوان بعد أن يقوم المعالج بإجراء الخطوتين التاليتين :

1-محتويات مسجل التجزء CS تتم إزاحتها ناحية اليسار بمقدار 4 بت فتصبح المحتويات الجديدة هي :

CS=12F00H

2-تجمع محتويات مؤشر الأوامر مع محتويات المسجل CS بعد الإزاحة فيتكون لدينا العنوان الحقيقى كالتالى :

$$\begin{array}{r} \text{C S} = 12\text{F}00\text{H} \\ \text{I P} = 001\text{B}\text{H} + \\ \hline \text{12F1BH} \end{array}$$

العنوان الحقيقى فى الذاكرة هو

أى أن الأمر الذى عليه الدور فى التنفيذ سيكون موجوداً فى الذاكرة فى العنوان 12F1BH (12F1BH كما رأينا فى المثال السابق . من ذلك نفهم حقيقة مهمة جداً وهي أن مؤشر الأوامر يشير أو يحدد عنوان فى الذاكرة منسوباً أو محسوباً بمحتويات المسجل CS . ولنضرب لذلك المثال التوضيحي التالي : افترض أن لدينا سيارة هنا فى القاهرة وأقصى ما تستطيع أن تقطعه هذه السيارة هو السير فى دائرة نصف قطرها 5 كيلومتر لتوزيع الحليب مثلاً ، هذه هى مقدرتها ! ... فهل تستطيع هذه السيارة أن توزع الحليب فى لندن ؟ نعم تستطيع إذا نقلناها إلى لندن بالطائرة ! إن هذه السيارة تقابل مؤشر الأوامر الذى يحتوى فقط 16 بت ولا يستطيع التعامل إلا مع 64 كيلو بايت فقط ولكن هذه 64 كيلو بايت تتحدد بدايتها بمحتويات المسجل CS بعد إزاحتها ، وبذلك فإن مؤشر الأوامر يستطيع جلب أى أمر من أى مكان فى الذاكرة التى تبلغ 1 ميجابايت بعد جمع محتوياته مع محتويات المسجل CS التى تمت إزاحتها لليسار ، تماماً مثل السيارة التى تستطيع أن توزع الحليب فى أى مكان فى العالم بعد نقلها بالطائرة للمكان المطلوب . لذلك فإنه فى بداية أى برنامج لابد من تحميل المسجل CS بالعنوان الذى نرغب فى كتابة البرنامج ابتداء منه وهذا العنوان بالطبع يكون فى أى مكان خلال الذاكرة التى تبلغ 1 ميجابايت . هنا يظهر سؤال وهو : لماذا ارتبط مؤشر الأوامر IP بالمسجل CS بالذات ولم يرتبط بأى واحد آخر من مسجلات

الجزء مثل المسجل DS أو SS مثلاً ؟ إن ذلك يرجع إلى الوظيفة المحددة لكل واحد من هذه المسجلات والتى نبينها فيما يلى :

1-4-1 مسجل تجزيء البرامح CS

يحتوى هذا المسجل عنوان بداية ذاكرة يبلغ 64 كيلو بايت يختص لكتابية شفرات البرامح فيه فقط ، ولذلك فإن مؤشر الأوامر يرتبط دائمًا بهذا المسجل لأن مؤشر الأوامر يشير على عنوان الأمر الذى عليه الدور فى التنفيذ ولابد أن الأمر يقع فى هذا الجزء من الذاكرة ، وينتدد العنوان الحقيقى للأمر كما ذكرنا بإضافة محتويات مؤشر الأوامر مع محتويات المسجل CS بعد إزاحتها 4 بت ناحية اليسار . يمكن أن تتغير محتويات المسجل CS فى أثناء تنفيذ البرنامج مع أوامر القفز أو النداء على البرامج الفرعية وذلك فى حالات خاصة سيأتي شرحها بعد ذلك .

1-4-2 مسجل تجزيء البيانات DS

يحتوى هذا المسجل على عنوان بداية ذاكرة جزء من الذاكرة يبلغ 64 كيلو بايت أيضًا وهذا الجزء يحتوى جميع البيانات التى يتعامل معها أو يحتاجها البرنامج ، تتم عنونة هذه البيانات بإضافة محتويات المسجل DS بعد إزاحتها لليسار 4 بت مع محتويات أى واحد من المسجلات DX أو BX أو SI أو DI كما سنرى عند دراستنا لطرق العنونة .

1-4-3 مسجل تجزيء المكادسة SS

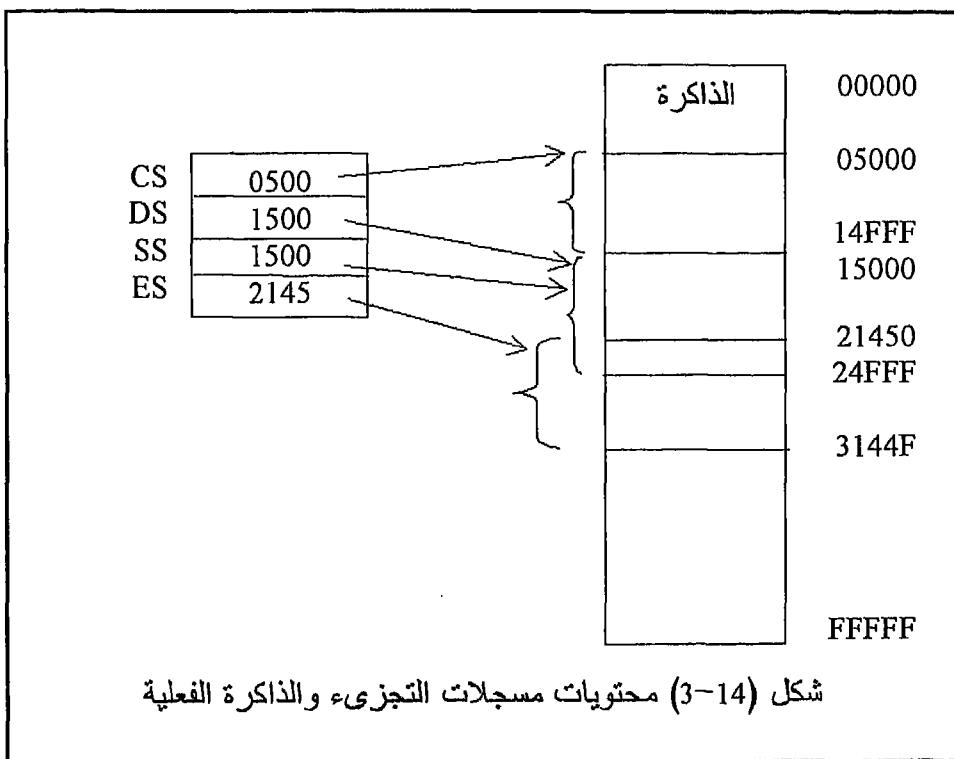
يحتوى هذا المسجل عنوان بداية ذاكرة يبلغ 64 كيلو بايت يستعملها المعالج كمكادسة . تستخد المكادسة لتخزين البيانات والعناوين الضرورية عند القفز إلى البرامح الفرعية أو القفز إلى برنامج لخدمة مقاطعة حيث من شأن هذه البيانات والعناوين التى تخزن فى المكادسة أن تساعد المعالج على الرجوع إلى نفس المكان الذى تم القفز منه فى البرنامج الأساسى واسترجاع القيمة الحقيقية لجميع المسجلات التى كانت موجودة قبل القفز بحيث يرجع المعالج إلى تنفيذ البرنامج الأساسى من حيث انتهى قبل القفز تماما دون خوف من تغير سير البرنامج بسبب فقد محتويات أحد المسجلات . يتم سحب البيانات من المكادسة على أساس أن آخر ما تم تسجيله يكون هو أول ما يتم سحبه أى أن آخر بايت تم تخزينها تكون أول بايت يتم سحبها Last In First Out, LIFO . تتم عملية السحب والإضافة فى المكادسة وبالتابع الذى أشرنا إليه بمساعدة مسجل مؤشر المكادسة Stack Pointer, SP حيث يحتوى هذا المسجل الذى يتكون من 16 بت على عنوان آخر مكان فى المكادسة تم التخزين فيه وبالطبع فإن مقدار المكادسة يتحدد ب 64 كيلو بايت كما ذكرنا . يتم تحديد العنوان الفعلى أو الحقيقى داخل هذا الجزء من الذاكرة (المكادسة) عن طريق إزاحة محتويات مسجل تجزيء

المكادسة SS ناحية اليسار بمقدار 4 برات ثم إضافة محتويات مؤشر المكادسة إليها.

4-4-4 مسجل التجزيء الإضافي ES Extra Segment register, ES

يحتوى هذا المسجل على عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت يستخدم لتخزين البيانات أيضا وبالذات سلاسل الحروف strings . يتحدد العنوان الحقيقي أو الفعلى لأى معلومة داخل هذا الجزء بازاحة محتويات مسجل التجزيء ES ناحية اليسار بمقدار 4 برات ثم يضاف إليه محتويات أى من المسجلين DI أو SI على حسب الأمر الذى يتم تنفيذه .

شكل (14-3) يبين محتويات مقتربة لكل واحد من مسجلات التجزيء والمساحة الفعلية التى يشغلها الجزء المقابل لكل مسجل على خريطة الذاكرة . نلاحظ من شكل (3ب) أن هذه الأجزاء يمكن أن تتدخّل مع بعضها البعض ، بل ويمكن أن تشغّل كلها نفس الجزء من خريطة الذاكرة .



4-4-5 مسجل الأعلام أو مسجل الحالة Status Register, SR

يحتوى هذا المسجل على 16 بت مستخدم منها 9 برات فقط كأعلام وباقى برات المسجل غير مستخدمه . إن كل بت أو علم من هذه الأعلام يعكس حالة معينة

من حالات نتيجة آخر عملية أو منطقية قام المعالج بتنفيذها . وفيما يلى نقدم هذه الأعلام والحالة التي يعكسها أو يبينها ومتى يكون كل علم صفرًا ومتى يكون واحدا على ضوء هذه النتيجة (سبق شرح معظم هذه الأعلام ولكن لا مانع من مراجعة سريعة لوظائفها) .

1- علم الحمل CF

إذا حصل حمل أو استلاف من أو إلى آخر بت نتيجة إجراء أي عملية حسابية أو منطقية فإن علم الحمل يصبح واحدا ، ويكون صفرًا فيما عدا ذلك . يتاثر هذا العلم أيضا ببعض أوامر الإزاحة والدوران .

2- علم الباريتي PF

إذا احتوت نتيجة آخر عملية أو منطقيةنفذها المعالج على عدد زوجي من الوهابيد فإن علم الباريتي يصبح واحدا ، أما إذا احتوت النتيجة على عدد فردی من الوهابيد فإن هذا العلم يصبح صفرًا .

3- علم الحمل النصفي HF

يكون هذا العلم واحدا إذا حصل هناك حمل أو استلاف من أو إلى البت الثالثة (منتصف البايت) عند إجراء أي عملية حسابية أو منطقية . لاحظ أننا نعد البتات في أي بايت ابتداء من الصفر ، أي البت رقم صفر ورقم واحد وهكذا .

4- علم الصفر ZF

يكون هذا العلم واحدا إذا كانت نتيجة آخر عملية حسابية أو منطقيةنفذها المعالج تساوى صفرًا ، ويكون هذا العلم صفرًا إذا كانت النتيجة تختلف عن الصفر .

5- علم الاشارة SF

يوضح هذا العلم إشارة آخر عملية حسابية أو منطقيةنفذها المعالج ، فإذا كانت هذه النتيجة سالبة يكون هذا العلم واحدا وإذا كانت هذه النتيجة موجبة فإن هذا العلم يكون صفرًا . لاحظ أن المعالج يعتبر النتيجة سالبة إذا كانت آخر بت فيها تساوى واحدا ويعتبر النتيجة موجبة إذا كانت آخر بت تساوى صفرًا . من ذلك نقول أن علم الإشارة يساوى دائمًا آخر بت في النتيجة .

6- علم الفخ أو المصيدة TF

هذا العلم لا يعكس نتيجة عمليةنفذها المعالج ولكنه حينما يكون واحد فإن المعالج ينفذ البرنامج خطوة بخطوة وحينما يكون صفر فإنه ينفذها بالطريقة المعتادة .

7- علم تنشيط المقاطعة IF

لكي يمكن مقاطعة المعالج 8086/8088 فإنه يتم إعطاؤه إشارة على طرف طلب المقاطعة INTR (الطرف 18 في شريحة المعالج) ولكن هذه المقاطعة لن يقبلاها المعالج إلا إذا كان علم المقاطعة IF فعال أي يساوى واحدا هو الآخر .

8-علم الاتجاه DF

هناك بعض الأوامر الخاصة بالتعامل مع سلاسل الحروف character strings من خلال المسجلين SI, DI حيث يزداد واحد أو ينقص واحد من محتويات هذين المسجلين ليشير إلى مكان معين في هذه السلسلة . يبين علم الاتجاه DF إذا كان سيكون هناك زيادة أم سيكون هناك نقص بمقدار واحد على محتويات هذين المسجلين . إذا كان $DF=1$ فإن ذلك يعني أنه سيكون هناك زيادة بمقدار واحد على محتويات هذين المسجلين ، وبالطبع إذا كان $DF=0$ فإن ذلك يعني أنه سيكون هناك إنفاص بمقدار واحد على هذه المحتويات .

9-علم الفيضان OF

يبين هذا العلم إذا كان هناك فيضان حسابي في نتيجة أي عملية حسابية مثل الجمع والطرح ألم لا . فمثلاً في حالة جمع الرقم 7FH الذي يساوى (127+) مع الرقم (1H) فإن النتيجة تكون 80H والتي تعتبر سالبة بعدأخذ المتم الثنائي لها وتساوي (-128) ، ولأن الرقم (-128) يعتبر غير صحيح لأن أي رقم سالب يجب ألا يتعدى (-127) فإن علم الفيضان يكون واحد . وعلى ذلك فإن هذا العلم يكون صفرًا طالما لم يكن هناك فيضان في النتيجة .

127	7FH
001 +	+01H
128-	80H

5-طرق العنونة Addressing modes

في أثناء تنفيذ المعالج لأى برنامج فإنه ينقل بيانات من مسجل إلى مسجل آخر أو من مسجل إلى مكان ما في الذاكرة أو من مكان ما في الذاكرة إلى أى مسجل داخل المعالج نفسه . هناك طرق مختلفة يمكن استخدامها لكي يتم ذلك وهذه الطرق المختلفة يجب أن يلم بها أى مبرمج حتى يكون برنامجه ذو كفاءة عالية . سنتقدم في هذا الجزء شرحًا مفصلاً لهذه الطرق المختلفة من خلال استخدام الأمر MOV كمثال تطبيقي . يقوم الأمر MOV بنقل معلومة من مكان (المصدر) وهذا المصدر يكون إما مسجل داخل المعالج نفسه أو بait من بيانات الذاكرة إلى مكان آخر (الهدف) وهذا الهدف أيضاً يكون إما مسجل أو مكان في الذاكرة . الصورة العامة لهذا الأمر هي :

MOV destination, source

حيث تنتقل المعلومة من المصدر إلى الهدف وكمثال على ذلك الأمر التالي :

MOV AX,BX

حيث تنتقل محتويات المسجل BX إلى المسجل AX وهو المركم . نلاحظ أنه دائمًا يكتب مصدر المعلومة بعد الفاصلة من ناحية اليمين وأما الهدف الذي ستنتقل إليه المعلومة فيكتب بجانب الأمر MOV وقبل الفاصلة .

1-5-1 عنونة المسجل

تستخدم هذه الطريقة لنقل معلومة (بأيت أو كلمة ، والكلمة 2 بait) من مسجل إلى مسجل آخر ، أى أن مصدر المعلومة يكون مسجلا وكذلك الهدف . وهذه الطريقة تعتبر أسرع الطرق لنقل معلومة من مكان إلى مكان حيث كل من مصدر وهدف المعلومة يكون مسجلا داخل المعالج نفسه ولا يتعامل المعالج مع الذاكرة على الإطلاق . كمثال على ذلك الأمران :

MOV CX,AX

الذى ينقل محتويات المسجل AX (16بت) إلى المسجل CX (16بت أيضًا) .

MOV AH,AL

الذى ينقل محتويات النصف الأول AL من المسجل AX إلى النصف الثاني AH فى المسجل نفسه .

من المهم جدا هنا أن نلاحظ أحجام المسجلات التى نتعامل معها ، فلا يصبح مثلا أن ننقل محتويات مسجل 8 بت إلى مسجل 16 بت أو العكس حيث سيعطى الأسمبلر رسالة خطأ على ذلك لأن ذلك غير مسموح . الجدير بالذكر هنا أنه فى مثل هذه الأوامر فإن مسجل المصدر لا تتغير محتوياته ولكن يؤخذ منها نسخة أو صورة وتوضع فى المسجل الهدف . فالأمر MOV CX,AX مثلا يأخذ نسخة من محتويات المسجل AX ويضعها فى المسجل CX دون تغير فى محتويات المسجل المصدر AX والذى يتغير فقط هو المسجل الهدف CX .

1-5-2 العنونة الفورية

Immediate addressing mode

تستخدم هذه الطريقة لنقل معلومة (بأيت أو كلمة) موجودة فى الأمر نفسه إلى مسجل من المسجلات . أى أن مصدر المعلومة هنا ليس مسجلا داخل المعالج ولا بait فى الذاكرة ولكن المعلومة تعتبر ثابت أو قيمة موجودة فى الأمر نفسه وبعد شفرة الأمر مباشرة ، كمثال على ذلك الأمر :

MOV AX, 34F6H

الذى يضع نسخة من الثابت أو الرقم أو المعلومة 34F6H المكونة من 16 بت وال موجودة فى البرنامج بعد شفرة الأمر MOV فى المسجل AX . لاحظ أن H فى آخر أي رقم تعنى أن هذا الرقم مكتوبا فى النظام السنتعشرى . بعض الأسمبلر تضع العلامة # أمام الثابت أو المعلومة الفورية ولكنها قليلة ونحن فى

هذا الكتاب لن نتبع ذلك وسنضع أى ثابت بدون هذه العلامة ، فقط سنضع حرف H للدلالة على أن الرقم ستعشرى أو إذا كان الرقم في النظام العشري فلن نضع أى علامة .

14-5-3 العنونة المباشرة Direct addressing mode

هنا يتعامل المعالج مع الذاكرة حيث سيرسل لها أو يستقبل منها معلومة ، وعلى ذلك لابد من تحديد عنوان هذه المعلومة . في العنونة المباشرة يحتوى الأمر نفسه على العنوان المباشر للمعلومة أو الثابت المراد جلبه أو إرساله من أو إلى الذاكرة . تذكر جيداً أن هذا العنوان يحدد نسبة إلى محتويات مسجل التجزئي DS ، فمثلاً الأمر MOV AL,[1234H] معناه نقل نسخة من محتويات العنوان 1234H في جزء البيانات إلى المسجل AL . لاحظ أنه بفرض أن محتويات المسجل DS=2000H فإن العنوان الفعلى للمعلومة السابقة سيكون 21234H بعد إزاحة محتويات المسجل DS لليسار 4 بت كما رأينا مسبقاً في حسابات العنوانين الفعليية . إذا كان العنوان الذي سيتم التعامل معه في جزء البيانات سيتكرر كثيراً في البرنامج فإنه يمكن في أول البرنامج إعطاء رمزاً لهذا العنوان ثم بعد ذلك يستخدم هذا الرمز للدلالة على هذا العنوان في أي مكان في البرنامج . فمثلاً يمكن أن نرمز للعنوان H1234 في المثال السابق بالرمز NUMBER باستخدام الأمر NUMBER EQU 1234H في أول البرنامج ، ثم بعد ذلك نستخدم الرمز MOV AL,NUMBER على هذا العنوان كما في الأمر NUMBER .

14-5-4 العنونة غير المباشرة Indirect addressing

هذه الطريقة من العنونة تسمح بالتعامل مع بيانات موجودة في الذاكرة حيث العنوان الذي سيتم التعامل معه في هذه الحالة يكون موجوداً في أحد مسجلات المعالج التالية: BX, BP, SI, DI . كمثال على ذلك افترض أن المسجل BX يحتوى الرقم 1000H وطلبنا من المعالج تنفيذ الأمر التالي: MOV AX,[BX] . في هذه الحالة سيقوم المعالج بإحضار نسخة من محتويات العنوان H1000H (والذى يليه) ويضعها في المسجل AX . أى أن محتويات المسجل BX الموضوع بين قوسين مربعين كما رأينا تمثل عنوان المعلومة وليس المعلومة نفسها ، ففي عدم وجود القوسين سينسخ المعالج محتويات المسجل BX ويضعها في المسجل AX كما رأينا في أول طرق العنونة (عنونة المسجل) . يجب أن نؤكد هنا أن العنوان الفعلى للمعلومة يحسب منسوباً لمحطيات مسجل التجزئي DS بعد إزاحتة ناحية اليسار 4 بتات كما ذكرنا سالفاً ، أى أنه إذا كانت محتويات المسجل DS=0100H فإن الأمر السابق سينسخ محتويات العنوان 01000+1000=02000H والذى يليه ويضعها في المسجل AX . لاحظ أيضاً أن

المسجلات BX, SI, DI تعنون عنوانين منسوبة إلى مسجل التجزيء DS بينما المسجل BP فيعنون عنوانين منسوبة لمسجل التجزيء SS . كامثلة على هذا النوع من العنونة انظر إلى الأوامر التالية :

```
MOV CX,[BX]  
MOV [BP],BL  
MOV [DI],AH  
MOV [DI],[BX] (خطأ)
```

حيث الأمر الأول سينقل محتويات عنوان (والذى يليه) فى جزء البيانات أو ذاكرة البيانات data segment المشار إليه بالمسجل BX إلى المسجل CX ، بينما الأمر الثانى سينقل محتويات النصف الأول من المسجل BX إلى عنوان مشار إليه بالمسجل BP ويقع فى جزء المكدة . الأمر الثالث سينقل النصف العلوى من المسجل AX إلى عنوان مشار إليه بالمسجل DI ويقع فى جزء البيانات ، أما الأمر الرابع فغير مسموح به لأنه ينقل من ذاكرة إلى ذاكرة وهذا النوع من العنونة غير مسموح به إلا فى حالات خاصة جدا مع بعض أوامر سلاسل الحروف .

5-5-14 عنونة القاعدة زائد الفهرسة Base plus index addressing

تعتبر هذه الطريقة من العنونة بمثابة عنونة غير مباشرة ولكن طريقة تكوين أو الحصول على العنوان تختلف عن الطريقة السابقة . هنا المسجلين BX و BP يستخدمان كقاعدة base أو كبداية لمجموعة أو صف أو موصولة array من العنوانين حيث BX تستخدم فى حالة وجود موصولة البيانات فى جزء البيانات و BP تستخدم فى حالة وجود موصولة البيانات فى جزء المكدة . فى هذا النوع من العنونة يتكون العنوان بجمع محتويات واحدة من مسجلات القاعدة BX أو BP مع محتويات واحد من مسجلات الفهرسة SI أو DI . يوضح ذلك المثال التالى :

```
MOV DL,[BX+DI]
```

حيث سيسخ المعالج محتويات العنوان المكون من جمع محتويات المسجلين BX و DI ويضعها فى النصف الأول من المسجل DX .

5-5-14 العنونة النسبية Relative addressing mode

هذا النوع من العنونة يختلف اختلافا بسيطا عن عنونة القاعدة زائد الفهرسة الذى تم شرحه سابقا حيث هنا يتم تحديد عنوان الذاكرة المراد التعامل معه عن طريق

جمع محتويات أحد المسجلات BX, BP, SI, DI مع إزاحة تعطى في الأمر نفسه
كما في المثال التالي :

MOV AX,[BX+1000]

حيث هنا سيتم عنونة العنوان المحدد بجمع محتويات المسجل BX مع الرقم 1000H وهذا العنوان سيكون في جزء البيانات من الذاكرة المحدد بمحتويات المسجل DS . جدول 14-1 يبين طرق العنونة السابقة مع مثال لكل طريقة ، حيث يمكنك مراجعته على ضوء ما سبق ويتأنى حتى يمكنك فهم هذه الطرق .

6- تمارين

1. ما هي المسجلات ذات 8 بت التي يمكن التعامل معها من خلال الأوامر للمعالج 8086/8088 ؟
2. قارن بين المعالجات 4 و 8 و 16 و 32 من حيث سرعة التنفيذ إذا تساوت كل العوامل الأخرى ؟
3. ما هي وحدة التنفيذ ووحدة مواجهة المسارات في المعالجين 8086/8088 ؟ وما أثرهما على أداء المعالج ؟ وما هو الفرق بين كل وحدة في كل من المعالجين ؟
4. ما هو طابور الإحضار Prefetch Queue ؟ وما هو تأثيره على أداء المعالج ؟ وكم عدد البيانات فيها في كل من المعالج 8086 و 8088 ؟
5. ما هي المسجلات ذات 16 بت ، والمسجلات ذات 8 بت التي يمكن التعامل معها من خلال الأوامر للمعالج 8086/8088 ؟
6. لماذا يطلق على المسجل CX Count register العد ؟ والمسجل DX Data register مسجل البيانات ؟
7. ما هو الخطأ في أوامر الانتقال التالية:

MOV AL,BX
MOV CS,SS
MOV ES,F214H
MOV [BX],[DI]

الأمر	العنوان الفعلى للمعلومة فى الذاكرة
MOV AL,BL	عنونة مسجل
MOV AL, temp	(10xDS)+temp
MOV AL,55H	عنونة فورية Immediate
MOV AL,[BP]	(10xSS) + BP
MOV AL,[BX]	(10xDS) + BX
MOV AL,[DI]	(10xDS) + DI
MOV AL,[SI]	(10xDS) + SI
MOV AL,[BP+5]	(10xSS) + BP + 5
MOV AI,[BX+4H]	(10xDS) + BX + 4H
MOV AL,[DI-66H]	(10xDS) + DI - 66H
MOV AL,[SI-400H]	(10xDS) + SI - 400H
MOV AL, temp[BX]	(10xDS) + temp + BX
MOV AL, temp[BP]	(10xSS) + temp + BP
MOV AL, temp[SI]	(10xDS) + temp + SI
MOV AL, temp[DI]	(10xDS) + temp + DI
MOV AL, temp[BX+10H]	(10xDS) + temp + BX + 10H
MOV AL, temp[BP-23H]	(10xSS) + temp + BP - 23H
MOV AL, temp[SI+50H]	(10xDS) + temp + SI + 50H
MOV AL, temp[DI+80H]	(10xDS) + temp + DI + 80H
MOV AL,[BX+DI]	(10xDS) + DI + BX
MOV AL,[BP+DI]	(10xSS) + DI + BP
MOV AL,[BX+SI]	(10xDS) + SI + BX
MOV AL,[BP+SI]	(10xSS) + SI + BP
MOV AL,[BX+DI+8]	(10xDS) + DI + BX + 8
MOV AL,[BP+DI-10H]	(10xSS) + DI + BP - 10H
MOV AL,[BX+SI-7]	(10xDS) + SI + BX - 7
MOV AL,[BP+SI+10H]	(10xSS) + SI + BP + 10H
MOV AL, temp[BX+SI]	(10xDS) + temp + BX + SI
MOV AL, temp[BP+SI]	(10xSS) + temp + BP + SI
MOV AL, temp[BX+DI]	(10xDS) + temp + BX + DI
MOV AL, temp[BP+DI]	(10xSS) + temp + BP + DI
MOV AL, temp[BX+SI+9]	(10xDS) + temp + BX + SI + 9
MOV AL, temp[BP+SI-10H]	(10xSS) + temp + BP + SI - 10H
MOV AL,temp[BX+DI+200H]	(10xDS) + temp + BX + DI + 200H
MOV AL, temp[BP+DI+1FH]	(10xSS) + temp + BP + DI + 1FH

جدول 1-14

8. أكتب أوامر الانتقال التي تقوم بالآتي:

- تحميل المسجل BX بالمعلومة 0F42H
- تحميل العنوان H 32000H بالمعلومة 0BH
- تصفير بaitات الذاكرة (أى جعل محتوياتها تساوى صفر) ابتداء من العنوان H 32000H إلى H 32050H بالتتابع
- نقل محتويات الذاكرة H 32000H حتى H 32050H إلى H 42000H حتى H 42050H
- تحميل المسجلات CS, SS, DS, ES بالعنوان H 3200H

9. ما معنى وضع التوسيع [] حول أى معامل من معاملات أى أمر ؟

10. ما هو عنوان الذاكرة الذى سيتم التعامل معه فى كل من الأوامر التالية إذا كانت DS=3200H, BX=0200H, DI=0300H, BP=1000H, SS=2000H و

: list=0250H

- MOV AL,[3200H]
- MOV AL,[BX]
- MOV [DI],AL
- MOV AL,[BX+100]
- MOV AL,[BP+100H]
- MOV AL,[BP+DI]
- MOV AL, list[DI]
- MOV AL, list[100H]
- MOV AL,[BX+DI]

الفصل الخامس عشر

برمجة المعالج

Intel 8086/8088

والمصحح

1-15 مقدمة

سنرى في هذا الفصل الخطوات الأولى في اتجاه كتابة برنامج بسيط بلغة الأسمبل الخاصة بالمعالج 8086/8088 ، ومن ثم تنفيذه ، كل ذلك من خلال برامج بسيطة نقدمها فقط لنفهم منها مكونات برمج لغة الأسمبل للمعالج 8086 . بعد ذلك يعرض هذا الفصل لمجموعات أوامر هذه اللغة عرضا سريعا الغرض منه هو التعريف بأهم مفردات هذه اللغة . بالطبع سيبقى هناك الكثير من الأوامر الأقل شيوعا ولكنها قد تفيد في الكثير من التطبيقات ولكننا لن نتعرض لها هنا ونحيل القارئ إلى أحد الكتب المتخصصة في لغة التجميع المذكورة في قائمة المراجع في نهاية هذا الكتاب . إن الأمور عادة لا تأتي بكل ما يتناء في البرمجة ، حيث كثيرا ما نجد أن البرنامج يحوي العديد من الأخطاء التي تعوق تنفيذ البرنامج بالصورة المطلوبة . سنرى في هذا الفصل أيضا بإذن الله كيفية استخدام برنامج `Debugger` أو المصحح "ديبجر" لاستخراج هذه الأخطاء والتخلص منها .

15-2 خطوات كتابة وتنفيذ برمج لغة التجميع

1. نبدأ بكتابه برنامج لغة الأسمبل مستخدمين الأوامر المختلفة لهذه اللغة كما سنرى تباعا بعد ذلك . يجب أن يتضمن البرنامج بعض الأوامر الموجهة للأسمبل لإخباره عن المتطلبات التي يحتاجها الأسمبل عند تحويل البرنامج إلى لغة الماكينة . وأول هذه الأوامر هو أمر إخبار الأسمبل عن مكان وضع البرنامج في الذاكرة مثلا ، وأيضا عن مكان وضع البيانات الناتجة عن البرنامج .

بعد الانتهاء من كتابة البرنامج يجب أن يخزن في ملف `file` بأي اسم مع مراعلة أن يكون امتداد هذا الملف `ASM` . ، فمثلا يمكن تسمية الملف بأي واحد من الأسماء التالية :

`Example.ASM`

`Test.ASM`

2. بعد ذلك يتم استدعاء الأسمبل وإدخال الملف الذي تم كتابته في الخطوة 1 عليه ، حيث سيعطينا الأسمبل نتيجة ذلك ملف جديد بنفس الاسم السابق ولكن بأمتداد مختلف ؛ هذا الملف سنسمي ملف الهدف `object file` وسيكون كالتالي :

`example.obj`

`Test.obj`

هذه الصورة من البرنامج تكون مكتوبة في صورة لغة الآلة ، ولكنها ما زالت غير مناسبة للتنفيذ بواسطة المعالج .

3. يتم بعد ذلك إدخال الملف السابق "ملف الهدف" على البرنامج الرابط linker الذي يقوم بتجميع الأجزاء المختلفة للبرنامج ، ووضعه في صورة مناسبة قابلة للتنفيذ executable بواسطة المعالج . هذه الصورة الجديدة للملف ستكون بنفس الاسم ولكن بامتداد جديد وهو EXE. وذلك كما يلي :

Example.EXE

Test.EXE

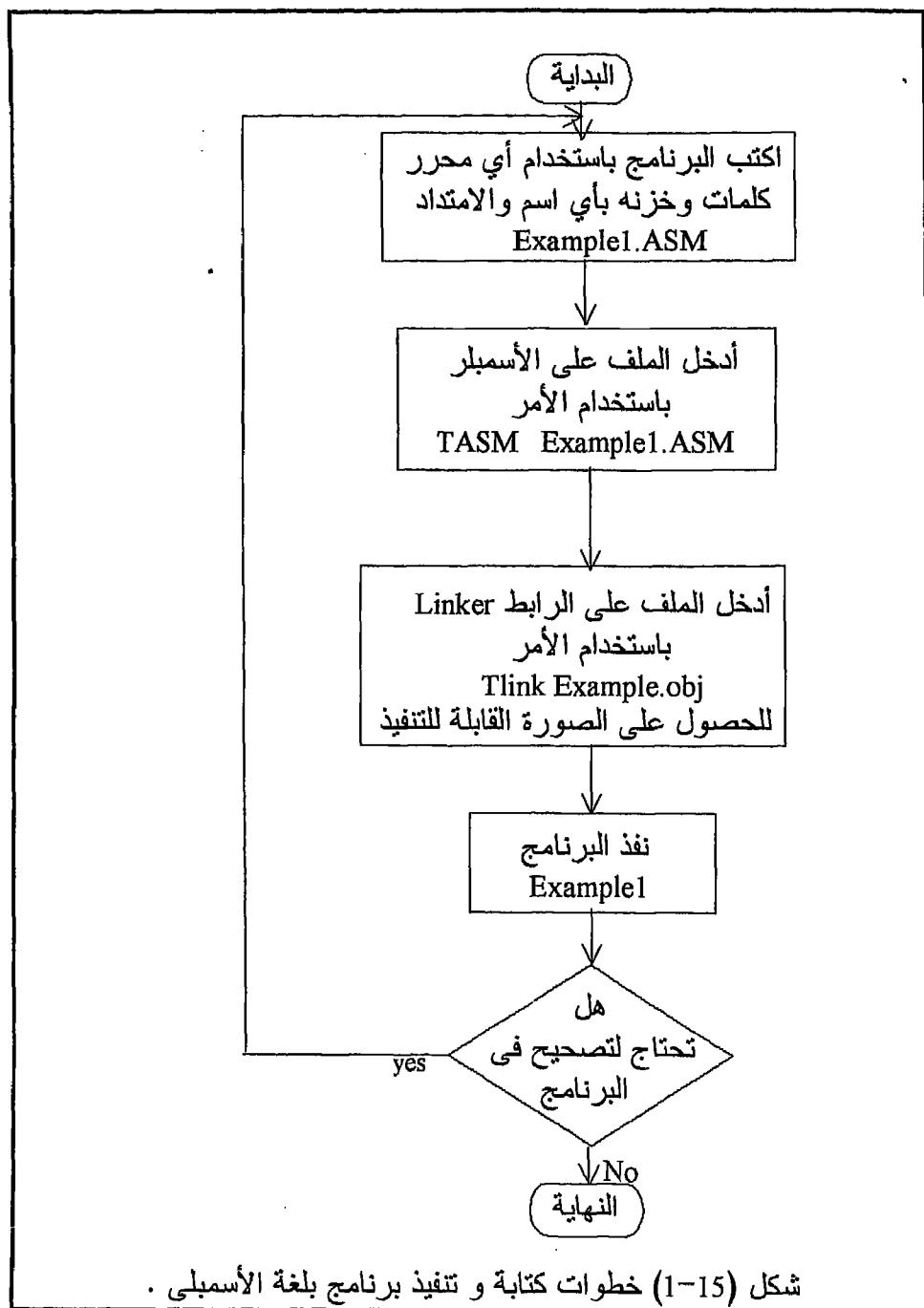
بعد الانتهاء من الخطوات الثلاث السابقة يمكن تنفيذ البرنامج ، وكذلك يمكن رؤية خرجه ، فإذا كان الخرج على ما يرام . . . تكون قد انتهينا من البرنامج ، أما إذا جاءت النتائج على خلاف ما نتوقع ، فإن ذلك يدل على وجود أخطاء في البرنامج . . فكيف يمكننا إذن الكشف عن هذه الأخطاء والتعامل معها ؟ إن هذا يتم عن طريق استخدام برنامج الدبجر ، وسنرى في هذا الفصل كيفية الدخول في هذا البرنامج واستخدامه . شكل (15-1) يبين رسمياً توضيحاً لكتابة برنامج بلغة الأسمبل ، وخطوات تنفيذه ، وذلك بفرض أن البرنامج تمت كتابته في ملف اسمه Example1.asm .

3-3 مكونات برنامج الأسمبل

لكي تتعرف على مكونات برنامج لغة الأسمبل ، سنسوق المثال الأول الذي يكتب الرسالة الآتية "آهلاً يا عرب ، استيقظوا" على الشاشة ، وذلك دون أن ننشغل بتفاصيل البرنامج الآن ، حيث سيرد ذكرها فيما بعد بإذن الله .

مثال 1-15

```
DOSSEG  
.MODEL SMALL  
.STACK 100H  
.DATA  
message DB 'Hello Arab , woke up ',B,10,'$'  
.CODE  
mov ax, @data  
mov ds,ax ; set ds to the beginning of the data segment
```



شكل (15-1) خطوات كتابة و تنفيذ برنامج بلغة الأسمبل .

mov ah,9 ; load register ah evith 9

```

mov dx , offset message
int 21h
mov ah , 4ch
int 21h
END

```

من هذا المثال نرى الآتي :

1. وجود مجموعة من الأوامر في أول البرنامج وهي عبارة عن أوامر توجيهية للasmblr directives . هذه الأوامر تخبر الأسمبلر عن حقائق معينة يريد المبرمج أن يأخذها في الاعتبار ، مثل تحديد جزء ذاكرة البيانات data segment ، وجزء ذاكرة البرنامج code segment ، وجزء ذاكرة المكدسة stack segment وكذلك نهاية البرنامج .

2. القسم الثاني من الأوامر هو أوامر لغة الأسمبلر مثل الأوامر mov و add و sub وغيرها ، وكلها عبارة عن أوامر سيقوم الأسمبلر بتحويلها إلى شفرات لغة الآلة و تخزينها في الذاكرة . لاحظ أن الأوامر التوجيهية التي سبق الإشارة إليها لا يتم تحويلها إلى شفرات لأنها ليس لها شفرات أصلًا ، وذلك لأنها ليست أوامر قابلة للتنفيذ بواسطة المعالج ، ولكنها مجرد توجيهات للأسمبلر لا يراها المعالج . فيما يلي سنأخذ فكرة موجزة عن أوامر التوجيه الموجودة في البرنامج السابق ، وهي كما يلي :

1- أمر التوجيه DOSSEG

هذا الأمر هو أول ما يكتب في أي برنامج أسمبلر ، وهذا الأمر يجعل كل أجزاء البرنامج (البيانات و الكود) تتبع نظام الميكروسوفت في التجزيء ، وهذا النظم لا يعنينا هنا في شيء ، ولن ننظر في أية تفاصيل أخرى له طالما أن هذا الأمر يقوم بهذه المهمة . المهم هنا هو أن نبدأ البرنامج بهذا الأمر كما ذكرنا .

2- أمر التوجيه MODEL

يحدد هذا الأمر للأسمبلر مodel الذاكرة الذي سيتم التعامل معه ، حيث تبعاً لهذا الموديل سيتحدد ما إذا كانت البيانات التي سيتعامل معها المعالج قريبة ؛ بحيث يتم عنونتها بـ 16 بت فقط داخل الجزء الخاص بها ، أم بعيدة فيتم التعامل معها على أساس 32 بت ، 16 منها تحدد العنوان داخل الجزء و 16 أخرى تحدد مكان أو بداية هذا الجزء . وهناك أكثر من موديل للذاكرة يتعامل معها الأسمبلر كما يلي :

1- الموديل tiny :

في هذا الموديل يكون البرنامج والبيانات موجودة في نفس الجزء حيث الجزء يبلغ 64 ك ب .

2- الموديل small :

يوجد كود البرنامج في جزء أو مقطع (64 ك ب) وبيانات البرنامج في جزء آخر منفصل عن الأول ولكن كلا من البرنامج والبيانات لا يتعدى الجزء الموجود فيه .

3- الموديل compact :

يوجد كود البرنامج في جزء معين ، أما بيانات البرنامج فيمكن أن تشغله أكثر من جزء واحد ، ولذلك فإن التعامل مع البيانات في هذه الحالة يكون على أساس أنها بعيدة ويكتب في 32 بت (segment : offset) بالرغم من أن البيانات هنا تشغله أكثر من جزء إلا أنه غير مسموح في هذا الموديل أن يكون لمصفوفة واحدة array أن تخرج خارج حدود هذا الجزء ، أي أن أي مصفوفة لا تتعدى 64 ك ب .

4- الموديل large :

هنا يمكن لكتابة البرنامج وبياناته أن يشغل كل منها أكثر من جزء واحد ، وسيكون التعامل مع العناوين هنا على الأساس البعيد سواء في حالة كود البرنامج أو بياناته . هنا أيضا يجب أن لا يتعدى حجم أي مصفوفة 64 ك.ب .

5- الموديل huge :

وهو يشبه تماما الموديل large في الوقت الحالي .
معظم البرامج التي نتعامل معها سواء في هذا المقرر أو في الكثير من التطبيقات الأخرى ، يكون الموديل small مناسبا جدا لها حيث أن البرنامج يكون مخصصا له 64 ك.ب ، وكذلك 64 ك.ب للبيانات ، وهذا يعني كافي جدا لهذه التطبيقات .
ويجب أن نستخدم هذا الموديل كلما أمكن إلا إذا كانت هناك ضرورة لغير ذلك لأن العنونة البعيدة الموجودة في الموديلات 3 ، 4 ، 5 تأخذ وقتا أطول في التنفيذ . وأخيرا يجب أن يوضع الأمر MODEL . قبل أوامر تحديد الأجزاء المختلفة .code و .data و .stack .

3- أمر التوجيه .stack

هذا الأمر يحدد كمية الذاكرة التي سيستخدمها البرنامج كمكادمة . والمكادمة يخزن فيها البرنامج عناوين الرجوع عند النداء على البرامج الفرعية ، أو تتفيد برامج خدمة المقاطعة . إن 200 كلمة تعتبر مناسبة جدا كمكادمة في الكثير من الأغراض ، حيث يتم تحديد هذه الكمية كما في الأمر التالي :
.stack 200h

4- أمر التوجيه .code

هذا الأمر يحدد بداية الجزء الذي سنكتب فيه شفرات أو كود البرنامج . لاحظ أن هذا الأمر ليس له معاملات تكتب بعده كما كان في الأمر `200h.stack`. ولكن هذا الجزء يبدأ بالأمر `code`. وينتهي بالأمر `END` ، ومثال على ذلك ما يلي :

```
.code
add ax,bx
sub ax,bx
mov cx,100
-----
END
```

5- أمر التوجيه .DATA

هذا الأمر يحدد بداية جزء البيانات المستخدمة في البرنامج كما يلي :

```
.DATA
boundary DW 100
counter DW 2
message DW ' ** ERROR MESSAGE ** ', '$'
-----
-----
```

6- أمر التوجيه END

بهذا الأمر تتحدد نهاية البرنامج ، وبدون هذا الأمر يعطي الأسمبلر رسالة خطأ ، لأنه من الضروري أن ينتهي البرنامج بهذا الأمر .

ملحوظة : إن نسيان بعض أوامر التوجيه السابقة يسبب خطأ ، وبعضها يسبب تحذير ، لذلك نؤكد على ضرورة الالتزام بها .

شكل (2-15) يبين الصورة العامة لبرنامج أسمبل و قد احتوى كل أوامر التوجيه السابقة .

```

DOSSEG
.MODEL SMALL
.STACK 100H
.DATA
DB
DW
-----
.Code
Your program

END

```

شكل (15-2) الصورة العامة لبرنامج الأسملي .

15-4 أوامر لغة الأسملي

لغة الأسملي للشريحة 8086/8088 تحتوى العديد من الأوامر بحيث أنه سيكون من الصعب ومن الملل جداً أن ندرس هذه الأوامر عن طريق سردها الواحد بعد الآخر إلى أن نصل إلى نهايتها بحيث عندما نصل إلى النهاية تكون قد نسينا ما درسناه في البداية . لذلك فقد اخترنا أن نقسم هذه الأوامر إلى مجموعات كما فعلنا عند دراسة لغة الأسملي للمعالجات السابقة بحيث ندرس كل مجموعة على حدة مع إعطاء بعض الأمثلة السريعة والتمارين على كل مجموعة ، معتمدين على أن الدارس لديه الخبرة الآن بمعظم أساسيات البرمجة بهذه اللغة .

15-5 مجموعة أوامر الانتقال Transfer instructions

هذه المجموعة من الأوامر خاصة بنقل البيانات من مكان لأخر دون إجراء أي تعديل أو تغيير عليها . الصورة العامة لهذه الأوامر هي :

`mov destination, source`

حيث الكلمة `mov` هي اختصار لكلمة `move` بمعنى أنقل أو حرك ، وأما `source` فهو مصدر المعلومة ، و `destination` هو الهدف أو الملجأ الذي تذهب إليه المعلومة . أي أن المعلومة ستنتقل من المصدر إلى الهدف . كل من المصدر

والهدف من الممكن أن يكون مسجلاً من مسجلات المعالج أو عنوان من عنوانين
الذاكرة . كما يمكن أن يكون المصدر معلومة فورية immediate أو ثابت .
من أمثلة نقل البيانات بين المسجلات المختلفة ما يلي :

نقل محتويات المسجل bl (8 بت) إلى المسجل al (8 بت) : mov al,bl ;

نقل محتويات المسجل cx (16 بت) إلى المسجل ax (16 بت) : mov ax,cx ;

نقل محتويات المسجل sp (16 بت) إلى المسجل bp (16 بت) : mov bp,sp ;

mov ds,ax

mov di,si

mov bx,es

هذا الأمر خطأ لأنه لا يمكن نقل محتويات مسجل مقطع إلى
إلى مسجل مقطع آخر ;

هذا الأمر خطأ لأن المسجلين أحدهما 16 بت والأخر 8 بت ; mov bl,ax ;

جميع الأوامر السابقة كانت تتعامل مع مسجلات فقط سواء كمصدر للمعلومة أو
هدف ستدهب إليه المعلومة . هذا هو ما يسمى أحياناً بعنونة المسجلات register
addressing حيث لا يكون هناك تعامل مع الذاكرة في طرفي الأمر ، فقط
مسجلات . يمكن تحويل أي مسجل بمعلومة فورية immediate data 8 بت أو
16 بت كما في الأوامر التالية :

mov al,03h

هذا الأمر يحمل النصف الأول من مسجل التراكم (8 بت) بالمعلومة الفورية 03h
(8 بت) ، حيث الحرف h يعني أن هذه المعلومة مكتوبة في النظام السعشرى .

mov ax,0ff35h

حيث هنا تم تحويل المسجل ax (16 بت) بالمعلومة ff35h (16 بت) . عادة يوضع
صفر قبل أي رقم سعشرى يبدأ بحرف كما في المثال السابق .

من المفيد جداً في الكثير من البرامج أن نرمز لقيمة فورية بأي رمز ثم نستخدم
هذا الرمز في البرنامج بدلاً من القيمة الثابتة ، كما في الأوامر التالية :

kkk equ 33h

.....

mov al,kkk

equ هو أمر توجيه جديد موجه للأسمبلر بإعطاء القيمة 33h للرمز kkk حيث
يقوم الأسمبلر باستبدال الرمز kkk بقيمتها عند كل موضع يظهر فيه هذا الرمز
في البرنامج .

مثال 2-15

أكتب برنامج يحمل المسجلات 00, 01, 02, 03, ah, al, bh, bl, ch, cl, dh بالقيم 04, 05, 06 على الترتيب ثم يقوم بعمل إزاحة دورانية على محتويات هذه المسجلات . هذا المثال تم تناوله مع كل المعالجات 8 بت ولذلك سنقدم البرنامج مباشرة كالتالي :

```
dosseg
.model small
.stack 100h
.data
.code
mov ah,00
mov al,01h
mov bh,02h
mov bl,03h
mov ch,04h
mov cl,05h
mov dh,06h
mov dl,dh
mov dh,cl
mov cl,ch
mov ch,bl
mov bl,bh
mov bh,al
mov al,ah
mov ah,dl
end
```

بعد كتابة هذا البرنامج سجله في ملف اسمه example1.asm وبعد ذلك استدعى الأسمبلر وأدخل عليه البرنامج باستخدام الأمر :

Tasm example1

للحصول على برنامج الهدف example1.obj . بعد ذلك استدعى برنامج التوصيل tlinker للحصول على الصورة القابلة للتنفيذ للبرنامج كما يلي :

tlink example1

بالحصول على الصورة القابلة للتنفيذ من البرنامج يمكنك تنفيذه باستخدام الأمر :

example1

حيث سينفذ البرنامج ويرجع الحاسب إلى dos دون أن ترى نتيجة محسوسة للبرنامج لأن مثل هذا البرنامج لا يطبع شيئاً على الشاشة ولا يرسل نتائج إلى الطابعة ، لذلك فلن تحس به لأنه فقط يغير من محتويات المسجلات داخل

المعالج . في مثل هذه الظروف يلعب الديبجر دوراً مهماً في أنه يمكننا به أن نرى نتيجة تنفيذ البرنامج في المسجلات ، حيث يمكن باستخدام الديبجر أن نفحص كل مسجلات المعالج لنرى محتوياتها بعد تنفيذ البرنامج لنعرف هل تم تنفيذ البرنامج بالطريقة المطلوبة أم لا . بل إنه من مزايا استخدام الديبجر أنه يمكننا تنفيذ البرنامج خطوة بخطوة لنرى نتيجة البرنامج بعد تنفيذ كل أمر ونفحص عند أي لحظة لنعرف هل البرنامج يسير على ما يرام أم لا . وهذه في الحقيقة تعتبر فائدة عظيمة في استخراج الأخطاء من البرامج . لذلك سنعرض في الجزء القادم لكيفية الدخول في البرنامج من الديبجر واستخدامه لتتبع تنفيذ البرنامج .

6-15 الديبجر Debugger

لكي تدخل في الديبجر لابد وأن يكون لديك الصورة القابلة للتنفيذ من البرنامج الذي تريد استخراج أخطاؤه أو التعامل معه ، لذلك يمكننا الدخول في الديبجر بالأمر التالي :

C:\TASM>debug example2.exe
 بذلك تدخل في الديبجر وتظهر لك علامة وجودك فيه حيث يصبح دليل الكتابة هو الشكل ‘-’، ويمكنك استخدام أوامر كال التالي : Cursor

15-1 اظهار محتويات المسجلات بالأمر R

بكتابة الحرف R (أو لأن لغة الأسمبل ليست حساسة لشكل الحرف) ثم enter ؛ تظهر أمامك جميع المسجلات بمحتوياتها كما يلى :

```
-r
AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=000
DS=272C ES=272C SS=273E CS=273C IP=0000 NV UP EI PL NZ NA PO NC
```

حيث نرى أن محتويات المسجل AX=0000 والمسجل CS=273C وهكذا . يمكنك إظهار محتويات مسجل معين بكتابة اسم المسجل بعد الحرف R حيث تظهر لك محتويات هذا المسجل فقط وفي السطر التالي تظهر العلامة ‘:’ والتي تتيح لك تغيير محتويات هذا المسجل بكتابة المحتويات الجديدة بعد هذه العلامة ، وإذا لم ترید تغيير هذه المحتويات اضرب enter .

يظهر في آخر قائمة المسجلات بيان حالة جميع الأعلام flags الموجودة في المعالج وحالة كل علم إذا كان واحد أم صفر . جدول (15-1) يبيّن قائمة بهذه

الأعلام وماذا يكتب فيها إذا كانت صفرًا وماذا يكتب في لها إذا كانت واحد .
سندرس معنى هذه الأعلام بالتفصيل عند دراستنا للققر المشروط .

اسم العلم	العلم مرفوع set to one	العلم غير مرفوع set to zero
Over flow	OV	NV
Direction	DN	UP
Interrupt	EI	DI
Sign	NG	PL
Zero	ZR	NZ
Auxiliary	AC	NA
Parity	PE	PO
Carry	CY	NC

جدول (15-1) بيان بحالة الأعلام التي يظهرها الدبيج

15-6-2 عرض أوامر الأسsembli ابتداء من عنوان معين Un

تحتوي الذاكرة الشفرات التثنائية لأوامر البرنامج ، وعرض هذه الشفرات التثنائية بنفس حالتها لا يفيد شيء حيث يكون من الصعب فهمها . لذلك فقد أتاح الدبيج إمكانية عرض هذه الأوامر بشفرات الأسsembli عن طريق كتابة الأمر Un والذى يعني عرض n من الأوامر ابتداء من العنوان الموجود فى المسجل CS كما يلى:

C:\TASM>debug example2.exe

```
-r
AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0000 NV UP EI PL NZ NA PO NC
18A8:0000 B400      MOV AH,00
-u0
18A8:0000 B400      MOV  AH,00
18A8:0002 B001      MOV  AL,01
18A8:0004 B702      MOV  BH,02
18A8:0006 B303      MOV  BL,03
18A8:0008 B504      MOV  CH,04
18A8:000A B105      MOV  CL,05
18A8:000C B606      MOV  DH,06
18A8:000E 8AD6      MOV  DL,DH
18A8:0010 8AF1      MOV  DH,CL
18A8:0012 8ACD      MOV  CL,CH
18A8:0014 8AEB      MOV  CH,BL
18A8:0016 8ADF      MOV  BL,BH
18A8:0018 8AF8      MOV  BH,AL
18A8:001A 8AC4      MOV  AL,AH
18A8:001C 8AE2      MOV  AH,DL
```

نلاحظ فيما سبق أن محتويات المسجل CS=18A8 ، لذلك تم عرض الأوامر ابتداء من العنوان 0000 نسبة لمحتويات هذا المسجل . بعد العنوان مباشرةً ستجد شفرة الأمر الستعرية ، فمثلاً الأمر MOV AH,00 كانت شفرته B400 والأمر MOV AH,DL شفرته هي 8AE2 .

15-3 عرض محتويات جزء من الذاكرة بالشفرات الستعرية بالأمر Dn

يمكن عرض محتويات جزء معين من الذاكرة بالأمر Dn حيث n هي عنوان البداية التي سيبدأ من عندها عرض المحتويات ، وعنوان البداية يكون هو العنوان الموجود في المسجل DS ، كما يلى :

```
C:\TASM>debug example2.exe
```

```
-r
```

```
AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0000 NVUP EI PL NZ NA PO NC
18A8:0000 B400      MOV    AH,00
```

```
-D0
```

```
1898:0000 CD 20 00 A0 00 9A F0 FE-1D F0 4F 03 FD 11 8A 03.....
```

```
1898:0010 FD 11 17 03 FD 11 EC 11-01 01 01 00 02 FF FF FF .....
```

```
1898:0020 FF FF FF FF FF FF FF-FF FF FF FF 83 18 4C 01
```

```
1898:0030 98 18-FF FF FF FF 00 00 00 00
```

```
-
```

لاحظ أن محتويات المسجل DS=1898 وتم عرض محتويات أماكن الذاكرة منسوبة لهذا العنوان وكل سطر يبين محتويات 16 عنوان (10 ستعرى) .

15-4 تنفيذ البرنامج حتى عنوان معين Ga

هذا الأمر يبدأ في تنفيذ البرنامج ابتدأ من العنوان المحدد بمسجل التجزيء CS ومحتويات مؤشر الأوامر IP . أي عنوان بداية التنفيذ سيكون CS:IP . سيفق التنفيذ عند العنوان a الموجود بعد الحرف G بحيث لن يتم تنفيذ الأمر الموجود عند هذا العنوان . لذلك لتنفيذ البرنامج السابق من بدايته لابد من تصغير المسجل IP أو لا استخدام الأمر RIP - ثم تحميل المسجل IP بأصفار . بذلك سيبدأ التنفيذ من العنوان 0000:1898 . بعد ذلك نعطيه أمر التنفيذ G001E - حيث

سيتوقف التنفيذ عند هذا الأمر الذي لا يدخل ضمن أوامر البرنامج ولذلك فإنه لن ينفذ .

15-6-5 متابعة تنفيذ البرنامج عن طريق تنفيذ عدد n من الخطوات

Tn

يمكن متابعة Trace تنفيذ البرنامج لاستخراج أخطاء التنفيذ عن طريق تنفيذه خطوة بخطوة باستخدام الأمر Tn حيث n هي عدد الأوامر المطلوب تنفيذها . فمثلاً T1 ستنفذ خطوة واحدة من البرنامج ، وهكذا . تذكر هنا أيضاً أن مؤشر الأوامر IP لابد أن يحتوى عنوان الخطوة المراد تنفيذها منسوباً لمسجل التجزيء CS . بعد تنفيذ كل خطوة يظهر الدبيجرا محتويات جميع المسجلات وكذلك الأمر التالي في التنفيذ كما يلي :

C:\TASM>debug example2.exe

r

AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0000 NVUP EI PL NZ NA PO NC
18A8:0000 B400 MOV AH,00

t1-

AX=0000 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0002 NVUP EI PL NZ NA PO NC
18A8:0002 B001 MOV AL,01

t1-

AX=0001 BX=0000 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0004 NVUP EI PL NZ NA PO NC
18A8:0004 B702 MOV BH,02

t2-

AX=0001 BX=0200 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0006 NVUP EI PL NZ NA PO NC
18A8:0006 B303 MOV BL,03

AX=0001 BX=0203 CX=001E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=1898 ES=1898 SS=18AA CS=18A8 IP=0008 NVUP EI PL NZ NA PO NC
18A8:0008 B504 MOV CH,04

15-6-6 تغيير محتويات عنوان في الذاكرة Ea

في الكثير من الأحيان يتطلب تنفيذ البرنامج وضع قيمة معينة في عنوان محدد في الذاكرة . يتم ذلك بالأمر Ea حيث a هي عنوان البايت المراد تغيير محتوياتها ، حيث بعد كتابة هذا الأمر يعرض الدبيجرا محتويات هذا العنوان الموجودة فعلياً وينتظر منك تغيير هذه القيمة .

15-7 الخروج من الدبيجر Q

بكتابة الحرف Q ثم enter تخرج من الدبيجر إلى دوس .
حاول كتابة مثال 1 الخاص بالإزاحة الدورانية لمحتويات المسجلات وجرب عليه كل أوامر الدبيجر .

15-7 تمارين

1. أشرح ما هو المقصود بالعنونة الضمنية ؟
2. ما هو المقصود بعنونة المسجلات ؟
3. أكتب باختصار عن الدبيجر ؟ وماذا تفعل لكي تتبع تنفيذ البرنامج خطوة بخطوة ؟
4. أكتب برنامج يحمل المسجلين AL , AH بأي بيانات ثم يقوم البرنامج باستبدال محتويات هذين المسجلين دون فقد محتويات أي مسجل منها ؟
5. أعد التمررين السابق ولكن على مسجلين 16 بت ، BX , AX مثلًا ؟
6. أعد برنامج الإزاحة الدورانية في مثال 1 ولكن هذه المرة أجعل الإزاحة تكون ناحية اليسار بدلاً من ناحية اليمين كما كان في المثال ؟
7. تتبع تنفيذ البرنامج التالي مع كتابة محتويات المسجلات بعد تنفيذ كل خطوة باعتبار أن جميع المسجلات تحتوى أصفار في بداية البرنامج :

	AH	AL	BH	BL	CH	CL	DH	DL
	00	00	00	00	00	00	00	00
MOV AL,05	--	--	--	--	--	--	--	--
MOV AH,01	--	--	--	--	--	--	--	--
MOV BX,AX	--	--	--	--	--	--	--	--
MOV CL,BH	--	--	--	--	--	--	--	--
MOV CH,BL	--	--	--	--	--	--	--	--
MOV DX,CX	--	--	--	--	--	--	--	--

8-15 أوامر القفز Jump Instructions

القاعدة العامة أن المعالج يقوم بتنفيذ البرنامج حسب ترتيب الأوامر من أول البرنامج حتى نهايته ، ولقد رأينا ذلك في الأمثلة السابقة . ولكن هناك بعض التطبيقات التي تتطلب الخروج على هذه القاعدة ، كأن يطلب منك مثلاً تنفيذ عملية معينة أو مجموعة من الأوامر عدد معين من المرات أو حتى عدد لا نهائي من المرات . لقد أتاح المعالج ذلك بتوفير بعض الأوامر التي تمكنك من برمجة من القفز بعملية التنفيذ من مكان لآخر خلال البرنامج . عادة تقسم أوامر القفز إلى نوعين كالتالي :

8-15-1 القفز غير المشروط jump Unconditional jump

عند تنفيذ أوامر القفز غير المشروط ينتقل المعالج بعملية التنفيذ إلى المكان الجديد والمحدد دون قيد أو شرط . هذا المكان الذي سيتم القفز إليه يكون محدداً بعلامة معينة label حيث تستخدم هذه العلامة في أمر القفز كذلك . هناك أمر وحيد للقفز غير المشروط وهو الأمر :

jmp label

كمثال على ذلك ما يلى :

```
again: mov ax,05H  
        mov bx,ax  
        mov cx,bx  
        jmp again
```

حيث العلامة again تم استخدامها قبل الأمر mov ax,05H المراد القفز إليه ، كما تم استخدامها أيضاً في أمر القفز نفسه . من شروط العلامات المستخدمة أنها لا بد أن تبدأ بحرف أبجدي ومن الممكن أن تحتوى أرقاماً ومن الممكن أن يصل عدد حروف العلامة إلى 31 حرفاً . يجب أيضاً أن لا تحتوى العلامة على مسافات ، ويجب أن تنتهي بالحرف ":" كدليل يبين نهاية العلامة . كذلك لا بد من وجود مسافة بين نهاية العلامة ":" وببداية الأمر .

8-15-2 القفز المشروط Conditional jump

كما يوحي الاسم فإن هذا النوع من القفز لن يتم إلا إذا تحقق شرطاً معيناً ، إذا لم يتحقق هذا الشرط فإن القفز لن يتم وسيستمر البرنامج في مساره الطبيعي حيث ينفذ الأمر التالي لأمر القفز . إن شروط القفز توضع دائماً على الأعلام ، فهناك

قفز مثلاً إذا كانت النتيجة تساوى صفراء ، أى أن علم الصفر يساوى واحد ، كما أن هناك قفزاً إذا كان هناك حملاً في آخر عملية قام بها المعالج ، وهكذا . جدول (15-2) يبين معظم أوامر القفز الشهيرة والكبيرة الاستخدام مع المعالج 8086/8088 . هناك أيضاً أوامر قفز مشروطة بحالة معينة لأكثر من علم مثل الأمر JA والذي يعني اقفز إذا كانت النتيجة أكبر من الصفر ، وتكون النتيجة أكبر من الصفر إذا كان علم الصفر يساوى صفراء وعلم الإشارة يساوى صفراء أيضاً .

أمر القفز	وصف الأمر
JA	اقفز إذا كانت النتيجة فوق الصفر jump if above
JAE	اقفز إذا كانت النتيجة فوق الصفر أو تساوى صفر jump if above or equal
JB	اقفز إذا كانت النتيجة تحت الصفر jump if below
JBE	اقفز إذا كانت النتيجة تحت الصفر أو تساويه jump if below or equal
JE/JZ	اقفز إذا كانت النتيجة تساوى الصفر jump if equal
JNC	اقفز إذا لم يكن هناك حمل jump if no carry
JNE	اقفز إذا كانت النتيجة لا تساوى الصفر jump if not equal
JNO	اقفز إذا لم يكن هناك فيضان في النتيجة jump if no overflow
JPO	اقفز إذا كانت الباريتى فردية jump if parity odd
JNS	اقفز إذا كانت النتيجة موجبة jump if no sign
JO	اقفز إذا كان هناك فيضان في النتيجة jump if overflow
JPE	اقفز إذا كانت الباريتى زوجية jump if parity even
JS	اقفز إذا كانت النتيجة سالبة jump if sign
JG	اقفز إذا كانت النتيجة أكبر من الصفر jump if greater
JGE	اقفز إذا كانت النتيجة أكبر من الصفر أو تساويه jump if greater or equal
JL	اقفز إذا كانت النتيجة أقل من الصفر jump if less
JLE	اقفز إذا كانت النتيجة أقل من أو تساوى jump if less or equal
JCXZ	اقفز إذا كان المسجل CX=0 يساوى صفر jump if CX=0

جدول (15-2) أوامر القفز المشروطة

مثال 3-15

أكتب برنامجاً يقارن محتويات المسجل AX والمسجل BX بحيث إذا كانوا متساويان يضع 1 في المسجل DX ، أما إذا كانوا غير متساوين فيوضع صفر في المسجل DX هذا مع الحفاظ على محتويات كل من المسجلين AX و BX .

أحد الاقتراحات لهذا البرنامج هو البرنامج التالي مع العلم أنه من الممكن أن يكون هناك أكثر من حل بالذات بعد أن ندرس باقي أوامر الحساب .

```
MOV CX,AX  
SUB AX,BX  
JZ HERE1  
MOV DX,0  
JMP HERE2  
HERE1: MOV DX,1  
HERE2: MOV AX,CX
```

نلاحظ أن هذا البرنامج قد احتفظ بمحطيات المسجل AX في المسجل CX كما في الأمر الأول ، بعد ذلك طرح محطيات المسجل BX من المسجل AX حيث ستوضع النتيجة في المسجل AX ، لذلك فإن محطيات المسجل AX ست فقد ولذلك فقد احتفظنا بها في المسجل CX . بعد ذلك إذا كانت نتائج الطرح صفراء فإن المسجلين متساوين وسيُضَع البرنامج 1 في المسجل DX بعد أن يقفز إلى العلامة HERE1 . أما إذا كان المسجلين غير متساوين فسيُضَع البرنامج صفر في المسجل DX ثم يقفز إلى العلامة HERE2 ليسترد محطيات المسجل AX من المسجل CX .

3-8-3 الأمر LOOP

من الأوامر الكثيرة الاستخدام لعمل الحلقات الأمر LOOP والذى ينفذ مجموعة الأوامر المحصورة بينه وبين العلامة المذكورة فيه عدد من المرات يساوى محطيات المسجل CX . كمثال على ذلك انظر إلى ما يلي :

```
MOV CX,10H  
HERE: MOV AX,00H  
.....  
.....  
LOOP HERE
```

حيث سينفذ هذا البرنامج مجموعة الأوامر الموجودة بين الأمر LOOP والعلامة HERE1 عدد 16 (10H) مرة حيث هذا العدد مخزن في المسجل CX قبل الدخول في البرنامج .

9-15 أول خطوات التعامل مع الذاكرة

First step to memory addressing

هناك طريقتان للتعامل مع الذاكرة وهما كما يلي :

15-1 الطريقة المباشرة Direct addressing

في هذه الطريقة يوجد العنوان المراد التعامل معه في الأمر نفسه مباشرة ، ولذلك سميت هذه الطريقة بالطريقة المباشرة للتعامل مع الذاكرة . كمثال على ذلك الأوامر التالية :

MOV al,[2000H]

وهذا الأمر يعني نقل محتويات البايت أو العنوان H 2000 في الذاكرة إلى المسجل al .

MOV AX,[2000H]

والذى يعني نقل محتويات العنوان H 2000 والذى يليه إلى المسجل AX .

MOV [31F2H],AL

والذى يعني نقل محتويات المسجل AL إلى البايت التي عنوانها H 31F2 .

نلاحظ أنه في كل هذه الأوامر ظهر العنوان مباشرة في الأمر نفسه ، ونلاحظ

أيضاً أن العنوان تم وضعه بين القوسين المربعين [] .

من الممكن أن يرمز للعنوان برمز معين كما في الأمر التالي :

MOV AL,table

والذى يعني نقل محتويات البايت المسمى بالاسم table إلى المسجل AL . إن

جميع العناوين المباشرة تكون دائماً محددة في مقطع أو جزء البيانات . أى أن

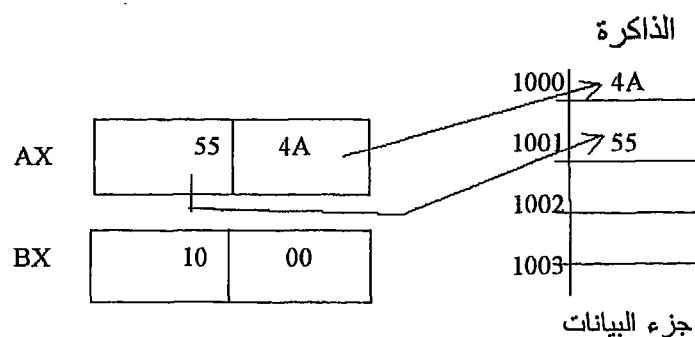
العنوان H 2000 مثلًا يحسب ابتداء من العنوان الموجود في مسجل التجزيء

. DS

15-2 الطريقة غير المباشرة Indirect addressing

هذه الطريقة من العنونة كما ذكرنا في الفصل الثاني تسمح بالتعامل مع بيانات موجودة في الذاكرة حيث العنوان الذي سيتم التعامل معه في هذه الحالة يكون موجوداً في أحد مسجلات المعالج التالية : BX, BP, SI, DI . كمثال على ذلك افترض أن المسجل BX يحتوى الرقم H 1000 وطلبنا من المعالج تنفيذ الأمر التالي : MOV AX,[BX] . في هذه الحالة سيقوم المعالج بإحضار نسخة من محتويات العنوان H 1000 (والذى يليه) الموجود في المسجل BX ويضعها في المسجل AX . أى أن محتويات المسجل BX الموضوع بين قوسين مربعين كما رأينا تمثل عنوان المعلومة وليس المعلومة نفسها ، ففي عدم وجود القوسين

سينسخ المعالج محتويات المسجل BX ويضعها في المسجل AX كما رأينا في أول طرق العنونة (عنونة المسجل) . يجب أن نؤكد هنا أن العنوان الفعلي للمعلومة يحسب منسوباً لمحتويات مسجل التجزيء DS بعد إزاحته ناحية اليسار 4 بتات كما ذكرنا سالفاً ، أى أنه إذا كانت محتويات المسجل DS=0100H فان الأمر السابق سينسخ محتويات العنوان $01000+1000=02000H$ والذى يليه ويضعها في المسجل AX . لاحظ أيضاً أن المسجلات BX, SI, DI تعنون عنوانين منسوبة إلى مسجل التجزيء DS بينما المسجل BP يعنون عنوانين منسوبة لمسجل التجزيء SS . شكل (3-15) عبارة عن رسم توضيحي لعملية العنونة غير المباشرة للذاكرة . كامثلة على هذا النوع من العنونة أيضاً انظر إلى الأوامر التالية :



شكل (3-15) العنونة غير مباشرة BX,ax

MOV CX,[BX]
 MOV [BP],BL
 MOV [DI],AH
 MOV [DI],[BX] (خطا)

حيث الأمر الأول سينقل محتويات عنوان (والذى يليه) فى جزء البيانات أو ذاكرة البيانات data segment والمضار إليه بالمسجل BX إلى المسجل CX . بينما الأمر الثاني سينقل محتويات النصف الأول من المسجل BX إلى عنوان مشار إليه بالمسجل BP ويقع فى جزء المكدسة . الأمر الثالث سينقل النصف العلوي من المسجل AX إلى عنوان مشار إليه بالمسجل DI ويقع فى جزء البيانات ، أما الأمر الثالث فغير مسموح به لأنه ينقل من ذاكرة إلى ذاكرة وهذا

النوع من العنونة غير مسموح به إلا في حالات خاصة جداً مع بعض أوامر سلاسل الحروف .

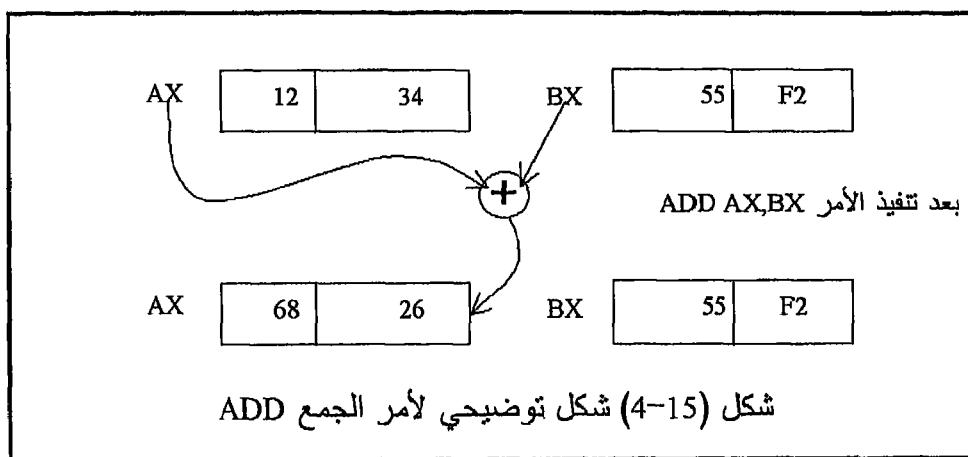
10-15 أوامر الحساب Arithmetic Instructions

أوامر الحساب التي يمكن تنفيذها بلغة الأسsembli هي الجمع والطرح والضرب والقسمة . كل هذه العمليات يمكن إجراؤها على بيانات 8 بت أو 16 بت .

1-10-15 أمر الجمع ADD

الصورة العامة لهذا الأمر هي :

ADD destination, source



حيث في هذا الأمر يتم جمع المصدر source مع الهدف destination وتوضع النتيجة في الهدف . من أمثلة ذلك ما يلي :

ADD AX,BX

هذا الأمر يجمع محتويات المسجل BX مع محتويات المسجل AX ويبقى النتيجة في المسجل AX . لاحظ أن هذا الأمر يجمع مسجلين كل منها 16 بت . شكل (4-15) يبين رسمياً توضيحاً لهذا الأمر .

أمثلة أخرى على هذا الأمر ما يلي :

ADD AL,CL

الذى يجمع محتويات النصف الأول من المسجل CX مع النصف الأول من المسجل AL وبضع النتيجة فى المسجل AL .

ADD AL,[BX]

الذى يجمع محتويات مكان الذاكرة (8بت) الذى عنوانه فى المسجل BX مع محتويات المسجل AL ويوضع النتيجة فى المسجل AL .

ADD AX,0F437H

هذا الأمر يجمع الثابت أو القيمة الفورية F437H (16بت) مع المسجل AX ويوضع النتيجة فى المسجل AX . فى العادة يتم وضع 0 قبل أي ثابت يبدأ بحرف وكذلك وضع الحرف H فى نهاية الثابت للدلالة على أنه فى النظام السعشرى .

ADD table,20H

الذى يجمع الثابت 20H مع محتويات مكان الذاكرة المسمى table ويوضع النتيجة فى نفس المكان . المكان المسمى table تم تسميته بهذا الاسم فى مقطع البيانات فى بداية البرنامج باستخدام أمر التوجيه define . كمثال على ذلك الأمر التالى :

• **table DB 22H**

هذا الأمر التوجيهى يحجز بait اسمها table ويعطىها القيمة الابتدائية 22 .

• **table DB 22H, 25H, 'A', 33H**

هذا الأمر يحجز عدد 5 بait ويعطىها القيم الابتدائية السابقة ، وهذه الخمسة أماكن تأخذ الاسم table . هناك أيضا الأوامر DW الذى يحجز كلمة word ، والأمر DD الذى يحجز كلمتين ، والأمر DQ الذى يحجز 4 كلمات ، والأمر DT الذى يحجز 10 كلمات . فى جميع هذه الأوامر الاسم الملحق بالأمر يمثل عنوان أول بait فى كمية الذاكرة المحجوزة .

من الأشياء المهمة التى يجب لا تنسى هي أنه لا يمكن جمع مكان ذاكرة على مكان ذاكرة آخر . فمثلا الأمر التالى خطأ فى عرف لغة الأسبللى :

ADD [BX],[3F20H]

لأنه يحاول جمع محتويات العنوان 3F20H مع محتويات العنوان الموجود فى المسجل BX وهذا خطأ أو غير مسموح .

10-15 أمر الجمع ADC

الصورة العامة لهذا الأمر هي :

ADC destination, source

حيث فى هذا الأمر يتم جمع المصدر source مع الهدف destination مع محتويات علم الحمل carry flag والتى تكون صفرًا أو واحدًا ، وتوضع النتيجة فى الهدف . من أمثلة ذلك ما يلى :

ADC AX,BX

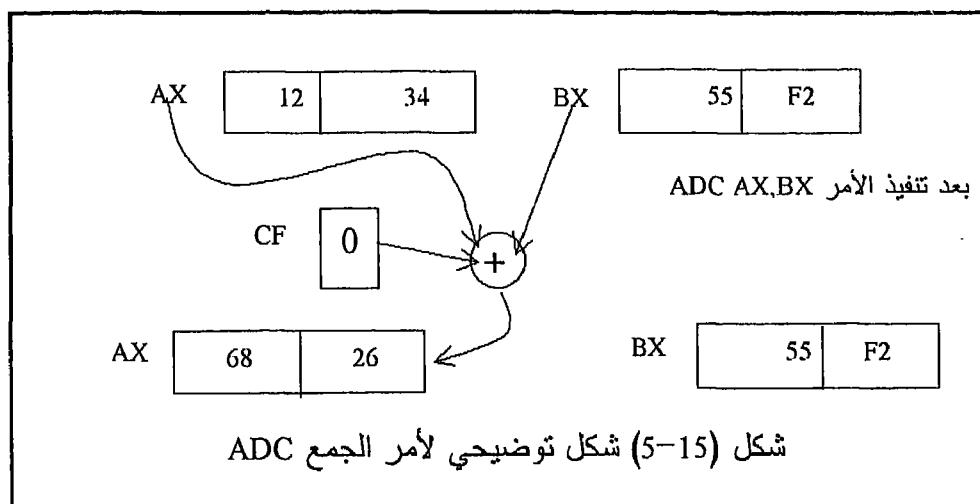
هذا الأمر يجمع محتويات المسجل BX مع محتويات المسجل AX مع محتويات علم الحمل ويوضع النتيجة في المسجل AX . لاحظ أن هذا الأمر يجمع مسجلين كل منها 16 بت مع علم الحمل . شكل (15-5) يبين رسمًا توضيحيًا لهذا الأمر . أمثلة أخرى على هذا الأمر ما يلي :

ADC AL,CL

ADC BL, table

ADC CL,40H

ADC BX, [SI]

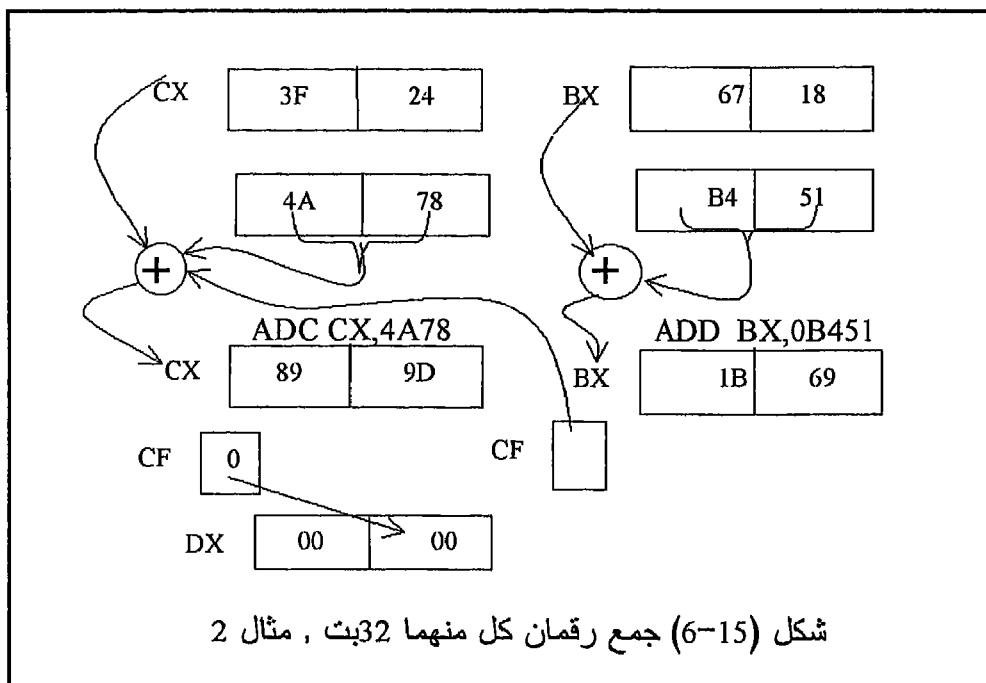


كل هذه الأوامر تجمع محتويات المصدر مع الهدف مع علم الحمل وتوضع النتيجة في الهدف . هنا يظهر سؤال مهم عن الفرق بين الأمرين ADD و ADC . لكنى نفهم الفرق بينهما نسوق المثال التالي :

مثال 4-15

اكتب برنامجاً يجمع الرقمين 3F246718 و 4A78B451 ويوضع النتيجة في المسجلات BX و CX و DX . نلاحظ أن كل من الرقمين مكون من 32 بت ، وليس هناك وسيلة لجمع رقمين كل منها 32 بت مرة واحدة . لذلك سنضع الرقم الأول في المسجلين BX و CX ثم نجمع النصف الأول من الرقم الثاني (B451) مع المسجل BX باستخدام الأمر ADD وبعد ذلك نجمع النصف الثاني من الرقم

الثاني (4A78) مع المسجل CX باستخدام الأمر ADC حتى يتمأخذ علم الحمل فى الاعتبار لأنه قد يكون هناك حمل من عملية الجمع الأولى فيجب أخذه فى الاعتبار . شكل(15-6) يبين رسمًا توضيحيًا لهذا المثال . لاحظ أن عملية الجمع الأولى يجب أن تتم باستخدام الأمر ADD وليس باستخدام الأمر ADC لأنه لو استخدم الأمر ADC فسيجمع علم الحمل من عملية سابقة مما سيؤثر على النتيجة و يجعلها خطأ .



شكل (15-6) جمع رقمان كل منها 32 بت ، مثال 2

البرنامج الذى سيقوم بعملية الجمع السابقة من الممكن أن يكون كالتالى :

ADD BX,0B451H
ADC CX,4A78H
MOV DX,00H
ADC DX,DX

10-3 أمر الطرح SUB

هناك أمران للطرح مثل الجمع ، أحدهما لا يأخذ علم الحمل فى الاعتبار والأخر يأخذ علم الحمل فى الاعتبار . الأمر SUB يطرح محتويات المصدر من محتويات الهدف ويوضع النتيجة في الهدف ، فمثلا الأمر :

SUB AX,BX

سيطرح المسجل BX (المصدر) من المسجل AX (الهدف) ويوضع النتيجة في المسجل AX . نؤكّد هنا على أن الهدف يكون هو دائمًا المطروح منه . أمثلة أخرى على أوامر الطرح كالتالي :

SUB DL,CL ;	DL-CL → DL
SUB AL,[BX] ;	AL-[BX] → AL
SUB BL,20H ;	BL-20 → BL

4-10-15 أمر الطرح SBB

هنا يتم طرح المصدر وعلم الحمل CF من الهدف وتوضع النتيجة في الهدف . من أمثلة ذلك ما يلي :

SBB AX,BX ;	AX-BX-CF → AX
SBB DL,CL ;	DL-CL-CF → DL
SBB AL,[BX] ;	AL-[BX]-CF → AL
SBB BL,20H ;	BL-20-CF → BL

يجب أن تكون حذرين جداً في الأماكن التي نستخدم فيها الأمر SUB والأماكن الأخرى التي يجب أن نستخدم فيها الأمر SBB لأن ذلك من الممكن أن يعطي نتيجة خاطئة كما أوضحتنا مع أوامر الجمع .

5-10-15 أمر المقارنة CMP

من الأوامر الشهيرة الاستخدام في الكثير من التطبيقات أمر المقارنة CMP الذي له الصورة العامة التالية :

CMP destination, source

حيث يتم طرح المصدر source من الهدف destination ولكن هنا لا يتم وضع النتيجة في الهدف ولكن يبقى الهدف دون تغيير ، وهذا هو الاختلاف الأساسي بين هذا الأمر وأوامر الطرح السابقة . الاستفادة الوحيدة من تنفيذ هذا الأمر هي تأثير الأعلام بنتيجة هذا الطرح . بالطبع ما أكثر التطبيقات التي تحتاج فيها لمقارنة رقمين لمعرفة أيهما أكبر من الآخر مثلاً دون تغيير قيمة أي واحد من الرقمين حيث في هذه الحالة فإن أوامر الطرح السابقة لا تؤدي هذا الغرض مباشرة ، لذلك في هذه الأحوال نستخدم الأمر CMP .

6-10-6 الأوامر DEC و INC

تعمل هذه الأوامر على زيادة محتويات المصدر أو إقصاصها بمقدار واحد .
الصورة العامة لهذين الأمرتين هي كالتالي :

INC source
DEC source

ومن أمثلة ذلك ما يلي :

INC BL
INC CX
INC [SI]
DEC DL
DEC [BX]

وجميعها تجمع 1 أو تطرح 1 من محتويات المصدر الموجود في الأمر سواء كان مسجل (8 أو 16 بت) أو بait ذاكرة .

مثال 5-15

أكتب برنامجا يقوم بضرب محتويات المسجلين CL و BL ويضع النتيجة في المسجل AX وذلك عن طريق الجمع المتكرر .

```

MOV AL,00
MOV AL,CL
DCR BL
YY: ADD AL,CL
JNC XX
INC AH
XX: DCR BL
JNZ YY
HLT

```

مثال 6-15

أكتب برنامجا يقسم محتويات المسجل AL على محتويات المسجل CL ويوضع النتيجة في المسجل AH والباقي في المسجل DL وذلك عن طريق الطرح المتكرر .

```

MOV AH,00
CMP AL,CL
JS XX;           CL>AL put AL into DL as a remainder
;           and stop
YY:  SUB AL,CL
JS XX1
INC AH
JMP YY
XX1: ADD AL,CL
XX:  MOV DL,AL ; put remainder in DL.
      HLT

```

10-7 أمر الضرب MUL

الصورة العامة لهذا الأمر هي :

MUL source

حيث يقوم هذا الأمر بضرب المصدر source الذى يكون مسجلا (8 بت أو 16 بت) أو بait ذاكرة فى المسجل AL و ذلك إذا كان المصدر 8 بت ، أما إذا كان المصدر 16 بت فإنه يضرب المصدر فى المسجل AX . أى أن الطرف الثانى لعملية الضرب يكون دائمًا إما المسجل AL أو المسجل AX على حسب عدد بิตات المصدر . إذا كان المصدر 8 بت فإن الطرف الثانى لعملية الضرب يكون المسجل AL كما ذكرنا وتوضع النتيجة فى المسجلين AL و AH حيث يحتوى AL النصف الأدنى من النتيجة ويحتوى المسجل AH النصف الأعلى منها . إذا كان المصدر 16 بت فإن الطرف الثانى لعملية الضرب يكون المسجل AX كما ذكرنا ، ولكن النتيجة فى هذه المرة توضع فى المسجلين AX (الذى يحتوى الجزء الأدنى من النتيجة) و DX (الذى يحتوى الجزء الأعلى منها) . بالطبع فإن جميع الأعلام تتأثر بهذه العملية . من أمثلة ذلك ما يلى :

MUL BL

الذى يضرب محتويات المسجل BL فى المسجل AL ويضع النتيجة فى المسجل . AX

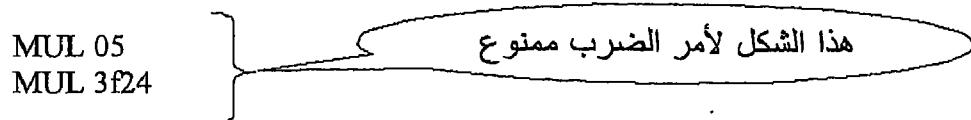
MUL CX

الذى يضرب محتويات المسجلين CX و AX ويضع النتيجة فى المسجلين AX و . DX

MUL [BX]

الذى يضرب محتويات بايت الذاكرة المشار إليها بالعنوان الموجود فى المسجل فى المسجل AL ويضع النتيجة فى المسجل BX .

من الأشياء المهمة التي يجب أن نحذرها أو نتجنبها هي ضرب قيمة فورية (ثابت) في المسجل AL أو AX على الصورة التالية :



في هذه الحالة يجب أن نضع الثابت في أي مسجل أولا ثم نستخدم هذا المسجل في عملية الضرب .

15-10-8 أمر القسمة DIV

الصورة العامة لهذا الأمر هي :

DIV source

حيث يقوم هذا الأمر بقسمة المسجل AX أو المسجلين AX و DX على المصدر source الذي يكون مسجلا (8 بت أو 16 بت) أو بait ذاكرة . أي أن المصدر

يكون دائما هو المقسم عليه . أما المقسم فيتعدد على حسب المصدر كالتالي :

1. إذا كان المصدر 8 بت فإن المقسم لا بد أن تكون بقائه ضعف بثات المقسم عليه ولذلك فإنه يكون المسجل AX في هذه الحالة . وفي هذه الحالة يخزن خارج القسمة في المسجل AL والباقي من عملية القسمة في المسجل AH .

2. إذا كان المصدر 16 بت فإن المقسم يكون المسجلين AX و DX حيث DX يحتوى النصف الأدنى و DX يحتوى النصف الأعلى للمقسم . أما ناتج القسمة فإنه يوضع في المسجل AX والباقي من القسمة فإنه يوضع في المسجل DX .

من أمثلة عمليات القسمة ما يلى :

DIV BL

الذى يقسم محتويات المسجل AX على المسجل BL ويضع النتيجة في المسجل AL والباقي في المسجل AH .

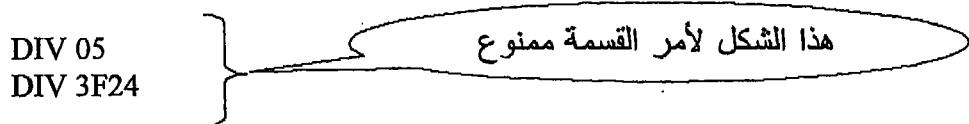
DIV CX

الذى يقسم محتويات المسجلين AX و DX على المسجل CX ويضع النتيجة في المسجل AX والباقي في المسجل DX .

DIV [BX]

الذى يقسم محتويات المسجل AX على بait الذاكرة المشار إليها بالعنوان الموجود في المسجل BX ويضع النتيجة في المسجل AL والباقي في المسجل AH .

من الأشياء المهمة التي يجب أن نحذرها أو نتجنبها هي استخدام قيمة فورية ثابت (ثابت) في عملية قسمة كالتالي :



في هذه الحالة يمكن أن يوضع الثابت في أي مسجل ثم يستخدم هذا المسجل في عملية القسمة .

مثال 7-15

أكتب برنامجاً يحسب مضروب الرقم الموجود في عنوان الذاكرة 1800H ويسكب نتيجة هذا المضروب في المسجل AX .

```
MOV AX, 01
MOV BL,[1800]
MOV BH,00
MOV CX,BX
DEC CX
XX: MUL BL
    DEC BL
    LOOP XX
    HLT
```

11-11 أوامر المنطق Logic Instructions

هذه المجموعة خاصة بإجراء العمليات المنطقية ، ويقصد بالعملية المنطقية العملية التي معاملاتها هي الواحد أو الصفر . العمليات المنطقية التي يجريها المعالج 8088 هي عمليات AND, OR, XOR, NOT .

11-11-1 الأمر AND
الصورة العامة لهذا الأمر هي :

AND destination, source

حيث يتم إجراء عملية AND على كل من المصدر source والهدف destination وتوضع النتيجة في الهدف destination . بالطبع فإن كل من المصدر والهدف لابد أن يكونا نفس الحجم أى لهما نفس العدد من البتات . من أمثلة ذلك ما يلي :
AND AL,BL

حيث يجري عملية AND على محتويات المسجلين AL, BL ويوضع النتيجة في المسجل AL . فمثلا افترض أن المسجلين AL, BL يحتويان البيانات التالية :

BL = 10111101=BDH

AL = 11101110=EEH

فإنه بعد تنفيذ الأمر AND AL,BL ستكون محتوياتهما كالتالي :

BL = 10111101=BDH

AL = 10101100=ACH

محتويات BL المصدر لم تتغير

النتيجة وضعت في الهدف

AND CX,DX

AND BH,table

AND CH,11011011B

حيث الأمر الأخير سيجري عملية AND على المسجل CH والمعلومة الفورية أو الثابت 11011011B المعطى في الصورة الثانية والتي تم الإشارة إليها باستخدام الحرف B في آخر الثابت وحيث تسمح لغة التجميع بذلك .

15-2-11-2 الأمر OR

الصورة العامة لهذا الأمر هي :

OR destination, source

حيث يتم إجراء عملية OR على كل من المصدر source والهدف destination وتوضع النتيجة في الهدف destination . بالطبع فإن كل من المصدر والهدف لابد أن يكونا نفس الحجم أى لهما نفس العدد من البتات . من أمثلة ذلك ما يلي :
OR AL,BL

حيث يجري عملية OR على محتويات المسجلين AL, BL ويوضع النتيجة في المسجل AL

OR CX,DX

OR BH,table
OR CH,11011011B

11-3 الأمر XOR

الصورة العامة لهذا الأمر هي :

XOR destination, source

حيث يتم إجراء عملية XOR على كل من المصدر والهدف وتوضع النتيجة في الهدف . من أمثلة ذلك ما يلي :

XOR AL,BL

حيث يجرى عملية XOR على محتويات المسجلين AL, BL ويوضع النتيجة في المسجل AL .

XOR CX,DX

XOR BH,table

XOR CH,11011011B

11-4 أمر النفي المنطقي NOT

هذا الأمر له معامل واحد والصورة العامة له هي كالتالي :

NOT source

حيث يتم عكس المصدر عكساً منطقياً يجعل كل صفر واحد وكل واحد صفر وبالطبع فإن نتيجة هذا العكس توضع في نفس المصدر . من أمثلة ذلك ما يلي :

NOT AL

NOT DX

NOT [BX]

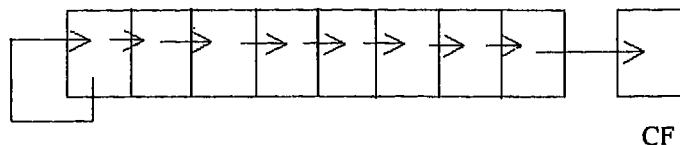
12-15 أوامر الإزاحة والدوران Shift And Rotate Instructions

هناك نوعان من الإزاحة ، إزاحة حسابية وهي التي يتم الاحتفاظ فيها بإشارة الرقم الذي تجري عليه الإزاحة وبالذات إذا كانت الإزاحة ناحية اليمين كما سرى . أما الإزاحة المنطقية فلا اعتبار فيها للإشارة .

12-1 أمر الإزاحة الحسابية لليمين SAR

يتم هنا إزاحة محتويات كل خانة إلى الخانة التي على يمينها ، وأما الخانة التي يتم تفريغها في أقصى اليسار فيتم ملأها دائمًا بمحتويات خانة الإشارة والتي هي نفس محتويات الخانة الأخيرة . أي أن محتويات آخر بت ناحية اليسار يتم

الاحتفاظ بها دائماً ولا تقرع . أما محتويات البت في أقصى اليمين فتذهب إلى علم الحمل . شكل (15-7) يبين رسمياً توضيحاً لهذا الأمر والصورة العامة له ستكون كالتالي :



شكل (15-7) الإزاحة الحسابية لليمين

SAR destination, count

حيث count إما أن تكون واحد إذا كانت الإزاحة ستم مرة واحدة فقط وأما إذا أردنا الإزاحة أكثر من مرة فيوضع عدد المرات في المسجل CL ثم يستخدم المسجل CL بدلاً من count في الصورة العامة للأمر . كمثال على ذلك نفترض أن المسجل $AL = 10011010$ بعد تنفيذ الأمر $SAR AL, 1$ فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :

$$AL = 10011010$$

$$AL = 11001101, \quad CF = 0$$

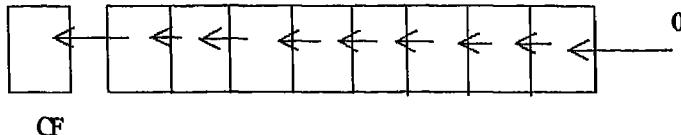
15-2 أمر الإزاحة الحسابية لليسار SAL

يتم هنا إزاحة محتويات كل خانة إلى الخانة التي على يسارها ، وأما الخانة التي يتم تفريغها في أقصى اليمين فيتم ملأها بأصفار . أما محتويات البت في أقصى اليسار فتذهب إلى علم الحمل . شكل (15-8) يبين رسمياً توضيحاً لهذا الأمر والصورة العامة له ستكون كالتالي :

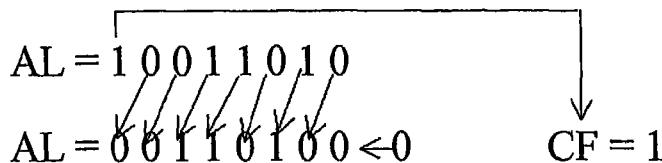
SAL destination, count

حيث count إما أن تكون واحد إذا كانت الإزاحة ستم مرة واحدة فقط وأما إذا أردنا الإزاحة أكثر من مرة فيوضع عدد المرات في المسجل CL ثم يستخدم المسجل CL بدلاً من count في الصورة العامة للأمر . كمثال على ذلك نفترض

أن المسجل $AL = 10011010$ بعد تنفيذ الأمر $SAL AL, 1$ فإن محتويات المسجل :
AL وعلم الحمل ستكون كالتالي :



شكل (15-8) الإزاحة الحسابية لليسار

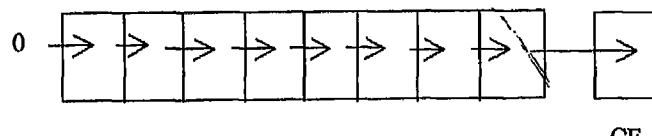


15-3 أمر الإزاحة المنطقية لليمين SHR

يتم هنا إزاحة محتويات كل خانة إلى الخانة التي على يمينها ، وأخر خانة ناحية اليسار يتم ملاؤها دائماً بـ 0 . أما محتويات البث في أقصى اليمين فتذهب إلى علم الحمل . شكل (15-9) يبين رسمياً توضيحاً لهذا الأمر والصورة العامة له ستكون كالتالي :

SHR destination, count

حيث count إما أن تكون واحد إذا كانت الإزاحة ستم مرة واحدة فقط وأما إذا أردنا الإزاحة أكثر من مرة فيوضع عدد المرات في المسجل CL ثم يستخدم المسجل CL بدلاً من count في الصورة العامة للأمر . كمثال على ذلك نفترض أن محتويات المسجل هي $AL = 10011010$ ، بعد تنفيذ الأمر $SHR AL, 1$ فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :



شكل (15-9) الإزاحة المنطقية لليمين

$$AL = 10011010$$

$$AL = 01001101, \quad CF = 0$$

15-12-4 أمر الإزاحة المنطقية لليسار SHL

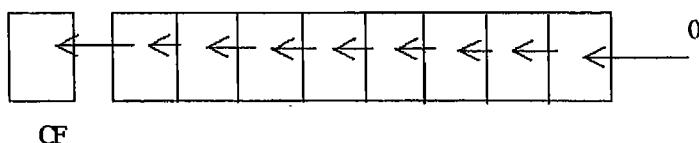
يتم هنا إزاحة محتويات كل خانة إلى الخانة التي على يسارها ، وأما الخانة التي يتم تفريغها في أقصى اليمين فيتم ملأها بأصفار . أما محتويات البت في أقصى اليسار فتذهب إلى علم الحمل تماما مثل الأمر SAL . شكل (15-10) يبين رسمياً توضيحيًا لهذا الأمر والصورة العامة له ستكون كالتالي :

SHL destination, count

حيث count إما أن تكون واحد إذا كانت الإزاحة ستم مرة واحدة فقط وأما إذا أردنا الإزاحة أكثر من مرة فيوضع عدد المرات في المسجل CL ثم يستخدم المسجل CL بدلاً من count في الصورة العامة للأمر . كمثال على ذلك نفترض أن المسجل $AL = 10011010$ بعد تنفيذ الأمر SAL AL,1 فإن محتويات المسجل وعلم الحمل ستكون كالتالي :

$$AL = 10011010$$

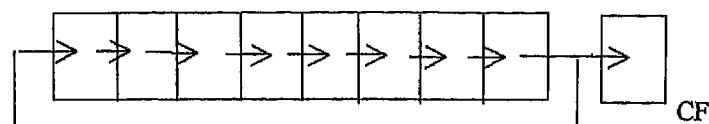
$$AL = 00110100 \leftarrow 0 \quad CF = 1$$



شكل (15-10) الإزاحة المنطقية لليسار

15-5 أمر الدوران لليمين ROR

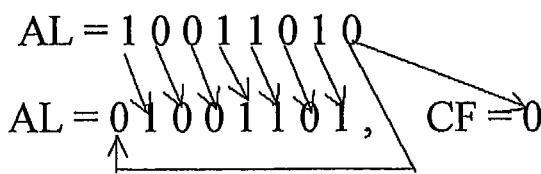
أوامر الدوران تشبه أوامر الإزاحة تماماً فيما عدا أن المحتويات التي ترافق من أي جهة يتم إدخالها مرة أخرى من الجهة الأخرى . شكل (15-11) يبين رسمياً توضيحاً لأمر الدوران لليمين والصورة العامة له ستكون كالتالي :



شكل (11-15) الدوران لليمين

ROR destination, count

حيث count هي كما شرحنا في الأوامر السابقة تماماً . كمثال على ذلك نفترض أن المسجل $AL = 10011010$ بعد تنفيذ الأمر $ROR AL, 1$ فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :

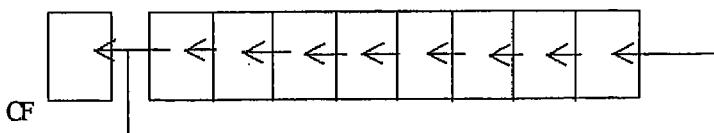


15-6 أمر الدوران لليسار ROL

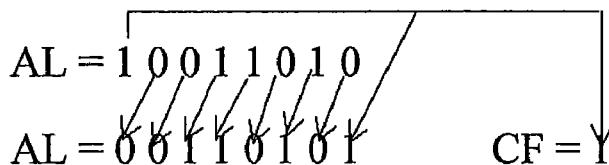
شكل (15-12) يبين رسمياً توضيحاً لهذا الأمر والصورة العامة له ستكون كالتالي :

ROL destination, count

كمثال على ذلك نفترض أن المسجل $AL = 10011010$ حيث بعد تنفيذ الأمر $ROL AL, 1$ فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي :



شكل (15-12) الدوران لليسار

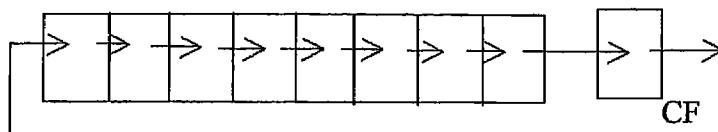


15-12-7 أمر الدوران لليمين من خلال علم الحمل RCR

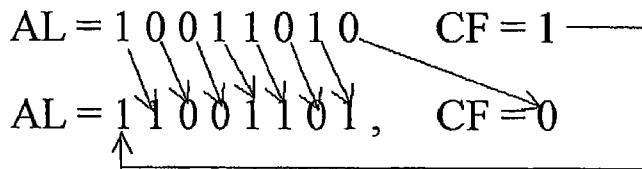
شكل (13-15) يبين رسمياً توضيحاً لأمر الدوران لليمين من خلال علم الحمل والصورة العامة له ستكون كالتالي :

RCR destination, count

حيث count هي كما شرحنا في الأوامر السابقة تماماً . كمثال على ذلك نفترض أن المسجل $AL = 10011010$ بعد تنفيذ الأمر RCR AL,1 فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي بافتراض أن علم الحمل كان فيه واحد قبل تنفيذ الأمر :



شكل (13-15) الدوران لليمين من خلال علم الحمل

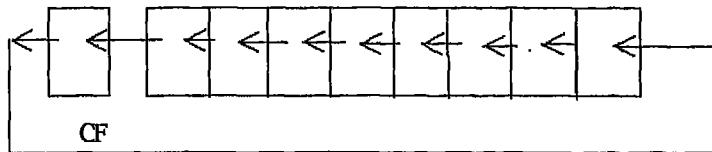


15-12-8 أمر الدوران لليسار من خلال علم الحمل RCL

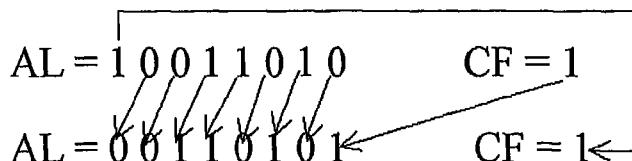
شكل (14-15) يبين رسمياً توضيحاً لهذا الأمر والصورة العامة له ستكون كالتالي :

RCL destination, count

كمثال على ذلك نفترض أن المسجل $AL = 10011010$ حيث بعد تنفيذ الأمر $RCL AL,1$ فإن محتويات المسجل AL وعلم الحمل ستكون كالتالي بافتراض أن علم الحمل كان واحدا قبل تنفيذ الأمر :



شكل (14-15) الدوران لليسار من خلال علم الحمل



مثال 8-15

أكتب برنامج يحسب مضروب الأرقام المخزنة في الذاكرة من $1800H$ إلى $1810H$ ويخزن المضروب في الأماكن $1900H$ إلى $1910H$. استخدم البرامج الفرعية معتبراً أن قيمة المضروب لن تزيد عن بait واحدة .

```

MOV DI,1800
MOV SI,1900
MOV CX,10H
MOV BH,00
XX: MOV BL,[DI]
PUSH CX
CALL FACT
POP CX
MOV [SI],AL
INC SI
MOV [SI],AH
INC SI
    
```

```

INC DI
LOOP XX
HLT
FACT: MOV AL,01
       MOV CX,BX
       DEC CX
YY:   MUL BL
       DEC BL
       LOOP YY
       RET

```

13-15 تمارين

1. هل يمكن جمع المسجلين CX و DS ؟
2. إذا كان المسجل CX=1001H والمسجل DX=20FFH فما هي محتويات كل من المسجلين وكذلك قيمة كل عل من الأعلام بعد تنفيذ أمر الجمع ADD عليهما؟
3. كرر السؤال السابق في حالة تنفيذ أمر الطرح SUB ؟
4. كرر السؤال الثاني في حالة إجراء الأوامر التالية AND و OR و XOR و NOT
5. أكتب برنامج يضع صفرًا في جميع أماكن الذاكرة 2000H إلى 20FFH ، وبعد ذلك يختبر نفس الأماكن ليرى إذا كان الصفر ما زال موجودًا أم لا ؟ هل يصلح هذا ليكون بمثابة اختبار لذاكرة الحاسب ؟
6. أكتب برنامج ينسخ محتويات بلوك الذاكرة 2000H إلى 20FFH في البلوك 3000H إلى 30FFH ؟
7. أكتب برنامجاً يختبر محتويات أماكن الذاكرة 1800H إلى 18FFH ويخرج منها الأعداد الفردية ويخزنها في الذاكرة من 1900H إلى 19FFH والأعداد الزوجية يخزنها في الذاكرة من 2000H إلى 20FFH .
8. أكتب برنامج يفحص محتويات الذاكرة من 2000H إلى 2500H ويبحث فيها عن عدد مرات تكرار الرقم 33H المكون من بait واحدة؟
9. أكتب برنامج يفحص محتويات الذاكرة من 2000H إلى 2500H ويبحث فيها عن عدد مرات تكرار الرقم 33FFH المكون من 2 bait؟
10. أكتب برنامجاً يحسب عدد الوحدات في كل بait من بaitات الذاكرة من 1800H إلى 18FFH ويخزنها في الذاكرة من 1900H إلى 19FFH .

11. أعدد السؤال السابق مستخدما البرامج الفرعية .
12. أكتب برنامجا يحسب مضروب الأرقام الموجودة فى الذاكرة من 1800H إلى 18FFH ويخزن هذا المضروب فى الذاكرة من 1900H إلى 19FFH . استخدم البرامج الفرعية .
13. أكتب برنامج يحسب عدد الأرقام السالبة وعدد الأرقام الموجبة فى المدى العنوانى من 18FFH - 1800H .
14. أكتب برنامج يقرأ محتويات المدى العنوانى 1700-17FFH ثم ينقل البيانات السالبة منها ويخزنها ابتداء من العنوان 1800H والبيانات الموجبة يخزنها ابتداء من العنوان 1900H .
15. أكتب برنامج يحسب عدد الأرقام الفردية وعدد الأرقام الزوجية فى المدى العنوانى 1700-17FFH .
16. أكتب برنامج يبحث عن أكبر رقم من بaitt واحدة فى المدى العنوانى 2000H إلى 20FFH ؟
17. أكتب برنامج يقوم بترتيب الأرقام (كل رقم بaitt واحدة) الموجودة فى المدى العنوانى 2000H إلى 3000H تحتوى بيانات إشارة صوتية ،
18. المدى العنوانى 2000H إلى 3000H يحتوى على بيانات إشارة صوتية ، والمطلوب حساب عدد مرات عبور هذه الإشارة لقيمة صفر ؟
19. استخدم البرامج الفرعية فى حساب المعادلة التالية :

$$M = 20! + 30! + 40! + 24^3 + 25^5 + 10^6$$

الفصل السادس عشر

مواجهة المعالج

8086/8088

1-16 مقدمة

لقد رأينا في الفصول السابقة كيفية مواجهة الشريحة 8085 مع الذاكرة ومع بوابات الإدخال والإخراج . إن مواجهة الشريحة 8086 لن تختلف كثيراً عن ذلك وسنحاول تقديمها هنا سريعاً من خلال شرح مختصر لوظيفة كل طرف من أطراف هذه الشريحة . ثم شرح عملية فصل المسارات المختلفة ، مسار العنلوين ومسار البيانات ومسار التحكم .

16-2 الوظائف المختلفة لأطراف الشريحة 8086/8088

كل من الشريحتين 8086 أو 8088 لها 40 طرفاً وهذه هي آخر أجيال شرائح المعالجات التي لها هذا العدد من الأطراف حيث كما سنرى في المعالجات القادمة أن عدد الأطراف قد قفز قفزة كبيرة . الاختلاف بين وظائف أطراف الشريحة 8086 والشريحة 8088 بسيطاً جداً وفي عدد محدود جداً من الأطراف كما سنرى .

قبل أن ندخل في تفاصيل وظائف هذه الأطراف يجب أن نعلم أن كل من الشريحتين تستخدم فكرة المزج الزمني time multiplexing التي سبق شرحها مع الشريحة 8085 بإسهاب ومع الكثير من الأطراف . إن فكرة المزج الزمني كما سبق وقدمناها تتلخص في أن نفس الطرف يمكن أن يحمل أكثر من إشارة في وقتين مختلفين . فمثلاً الطرف AD0 يحمل إشارة عنوانين وإشارة بيانات أيضاً ، فهذا الخط يمثل الإشارة D0 عند لحظة معينة ، كما يمثل الإشارة A0 عند لحظة أخرى . قبل مواجهة شريحة المعالج مع آية شريحة أخرى لابد من فصل إشارة البيانات على خط منفصل وإشارة العنوانين على خط آخر . تتم عملية الفصل باستخدام الطرف ALE الذي يكون واحداً عندما تكون الإشارة على الطرف AD0 مثلًا تتمثل عنوانين ، ويكون صفرًا عندما تكون الإشارة على هذا الخط تمثل بيانات .

هنا نرجو من القارئ أن يراجع الفصل الخاص بفصل أو عزل مسارات الشريحة 8085 حيث أنها هي نفسها بالضبط المستخدمة مع الشريحة 8086/8088 .
شكل (16-1) يبين رسمياً طرفيًا لكل من الشريحتين 8086/8088 حيث يمكن عرض وظائف هذه الأطراف كما يلي :

1. الخطوط AD0 إلى AD7 تحمل مزيج من إشارة العناوين و البيانات حيث تكون الإشارة على هذه الأطراف عناوين عندما يكون الطرف ALE فعال أي واحد ، وتكون بيانات عندما يكون الطرف ALE غير فعال أي صفر . هذا الكلام مطبق على كل من الشريحتين 8086 أو 8088 .

GND	1	40	Vcc	GND	1	40	Vcc
AD14	2	39	AD15	A14	2	39	A15
AD13	3	38	A16/S3	A13	3	38	A16/S3
AD12	4	37	A17/S4	A12	4	37	A17/S4
AD11	5	36	A18/S5	A11	5	36	A18/S5
AD10	6	35	A19/S6	A10	6	35	A19/S6
AD9	7	34	BHE/S7	A9	7	34	SS0
AD8	8	33	MN/MX	A8	8	33	MN/MX
AD7	9	32	RD	AD7	9	32	RD
AD6	10	31	HOLD	AD6	10	31	HOLD
ADS	11	30	HOLDA	AD5	11	30	HLDA
AD4	12	29	WR	AD4	12	29	WR
AD3	13	28	M/I/O	AD3	13	28	IO/M
AD2	14	27	DT/R	AD2	14	27	DT/R
ADI	15	26	DEN	ADI	15	26	DEN
AD0	16	25	ALE	AD0	16	25	ALE
NMI	17	24	INTA	NMI	17	24	INTA
INTR	18	23	TEST	INTR	18	23	TEST
CLK	19	22	READY	CLK	19	22	READY
GND	20	21	RESET	GND	20	21	RESET

أ- الشريحة 8086

ب- الشريحة 8088

شكل (1-16) الرسم الطرفي للمعالجين 8086/8088 في الحالة
الحقيقية أو الصغرى
• Minimum mode

2. الخطوط AD8 إلى AD15 ، في حالة الشريحة 8088 تسمى هذه الأطراف A8-A15 حيث أنها في هذه الحالة تحمل إشارة عناوين فقط طول الوقت وليس هناك أي مزج زمني في الإشارات لأن مسار البيانات في هذه الشريحة 8 خطوط فقط وينتهي عند الطرف AD7 . أما في حالة الشريحة 8086 فإن مسار البيانات يكون 16 طرفا ، لذلك فإن الخطوط AD8-AD15 تحمل مزيجاً من إشارة البيانات D8 إلى D15 وإشارة العناوين A8 إلى A15 حيث تكون الإشارة على هذه الخطوط إشارة عناوين عندما يكون الطرف ALE فعالا (1) ، وتكون

الإشارة بيانات عندما يكون الطرف ALE غير فعال (0) كما في حالة الخطوط . AD0-AD7

3. الأطراف A16/S3 و A17/S4 و A18/S5 و A19/S6 تحمل إشارة عنوانين A16-A19 حينما يكون الخط ALE فعالا . عندما يكون الخط ALE غير فعال فإن هذه الخطوط تحمل الإشارات S3, S4, S5, S6 التي تمثل حالات مختلفة للمعالج . حيث الطرف S6 يكون صفرًا دائمًا ، والطرف S5 يبين حالة علم المقاطعة ، والطرفين S3, S4 يبيّنان أي مقطع من الذاكرة يتم التعامل معه في حالة التعامل مع الذاكرة كما في جدول 1-16 .

S4	S3	الوظيفة
0	0	Extra segment
0	1	Stack segment
1	0	Code
1	1	Data

جدول 1-16

4. الطرف RD هذا الطرف يكون فعالا (0) حينما يكون المعالج في حالة قراءة للبيانات سواء من الذاكرة أو من بوابة إدخال . أي أن البيانات الموجوّدة على مسار البيانات تكون داخلة للمعالج ولا يهم من أي مصدر تكون هذه البيانات .

5. الطرف READY . لكي يقوم المعالج بتنفيذ أي أمر ، لابد وأن يكون هذا الطرف فعالا (1) . إذا أصبح هذا الخط صفرًا فإن المعالج يدخل في حالة الانتظار ، حيث تجمد جميع أطراف المعالج على الحالة التي كان عليها ، ويستفاد بذلك عند مواجهة المعالج مع بعض الأجهزة الخارجية البطيئة مثل شوائط الذاكرة البطيئة أو أجهزة الإدخال والإخراج مثل الطابعات والتي لا تضاهي سرعتها سرعة المعالج .

6. الطرف INTR ، حينما يكون هذا الطرف فعالا (1) تتم مقاطعة المعالج ، حيث يكمل تنفيذ الأمر المشغول به ثم يذهب لتنفيذ برنامج معين لخدمة هذه المقاطعة ، وبعد الانتهاء من هذه الخدمة يرجع المعالج إلى نفس مكان البرنامج الأساسي الذي تمت عنده المقاطعة .

7. الطرف TEST ، الأمر WAIT والذي يمثل حالة الانتظار يدخل المعالج فيها ويظل فيها طالما أن الخط TEST غير فعال (0) . عندما يصبح هذا الخط فعالا (0) فإن المعالج يخرج من حالة الانتظار ويستمر في تنفيذ البرنامج الأساسي . حالة الانتظار هنا يتم الدخول فيها بالأمر WAIT على العكس من حالة الانتظار

- التي يتم الدخول فيها بجعل الخط READY صفرًا ، حيث تنتهي هذه الحالة برجوع الخط READY واحد مرة أخرى .
8. الطرف NMI ، وهو طرف المقاطعة الغير قابلة للحجب أو غير المقنعة ، حيث تتم مقاطعة المعالج هنا بانتقال الخط NMI من الصفر إلى الواحد (أي عند الحافة الصاعدة) ، ولائيهم هنا أن يكون علم المقاطعة يساوي واحد ، وتوضع على هذا الطرف إشارات المقاطعة المهمة مثل إشارات انقطاع القدرة مثلاً .
9. الطرف RESET ، أو إعادة الوضع ، حيث عندما يكون هذا الطرف واحد فإن المعالج يذهب فوراً إلى عنوان الذاكرة FFF0H ويببدأ التنفيذ من هناك . وفي العادة يكون عند هذا العنوان أمر قفز إلى مكان آخر في الذاكرة يحتوي شفرة برنامج خاص بإعادة الوضع للمعالج .
10. الطرف CLK يتم إدخال نبضات التزامن من على هذا الطرف بالتردد المطلوب و duty cycle أو زمن ON/OFF بنسبة 33% حتى نضمن التزامن المطلوب لجميع العمليات التي يتقدّمها المعالج .
11. الطرف Vcc حيث يوضع جهد القدرة الثابت $5V \pm 10\%$.
12. الطرف GND وهو طرفاً يجب أن يكونا متصلين بباقي أطراف الأرضى في النظام الخارجي .
13. الطرف MN/MX ، وهو طرف الحالة حيث عندما يكون هذا الطرف (1) فإن المعالج يعمل في الحالة الصغرى Minimum ، وإذا كان هذا الطرف (0) فإن المعالج يعمل في الحالة العظمى Maximum . الحالة الصغرى هي الحالة العادية للمعالج 8086 ، وأما الحالة العظمى فهي حالة المعالج عندما يتعامل مع معالج العمليات الحاسوبية 8087 .
14. الطرف BHE/S7 يستخدم هذا الخط لتنشيط البایت ذات القيمة العظمى من مسار البيانات D0-D15 في أثناء قراءة أو كتابة البيانات . هذه الإشارة تمتزج مع إشارة الحالة S7 . هذا الطرف بالطبع موجود فقط في المعالج 8086 الذي يتعامل على أساس أن البيانات الخارجية 16 بت .
15. الطرف IO/M في المعالج 8088 هذا الطرف يبيّن ما إذا كانت الإشارة الموجودة على مسار العناوين تمثل عنوان في ذاكرة أم عنوان لوحدة إدخال أو إخراج . هذا الخط يكون (1) إذا كانت الإشارة تمثل عنوان في ذاكرة أو وحدة إدخال أو إخراج ويكون (0) إذا كانت الإشارة تمثل عنوان في ذاكرة . لاحظ وجود شرطة على الحرف M في اسم هذا الطرف . في حالة المعالج 8086 تكون الإشارة على هذا الطرف معكروسة ورمز الخط في هذه الحالة هو M/IO .
16. الطرف WR مثل الطرف RD يكون فعالاً (0) في حالة كتابة بيانات في الذاكرة أو أي وحدة إخراج .

17. الطرف INTA ، عند الاستجابة لطلب المقاطعة على الطرف INTR ، فإن المعالج يجعل الخط INTA فعالا (0) للدلالة على أنه قبل المقاطعة واعترف بها . وينتظر إدخال رقم المقاطعة التي سيقوم بخدمتها على مسار البيانات .
18. الطرف ALE ، وهو خط مسك العنوان Address latch enable يستخدم للدلالة على أن الإشارة الموجودة على الخطوط AD0-AD15 تمثل عنوانين أم بيانات . عندما يكون هذا الخط واحد فإن الإشارة على هذه الخطوط تمثل عنوان وعندما يكون الخط ALE صفرًا فإن الإشارة على هذه الخطوط تمثل بيانات . يستخدم هذا الخط لفصل إشارة العنوانين عن البيانات كما سنرى .
19. الخط DT/R ، خط إرسال أو استقبال البيانات حيث يستخدم هذا الخط لبيان إذا كانت البيانات خارجة T أو داخلة Transmit، R أو Receive للمعالج . لذلك فإن هذا الخط سيستخدم لتحديد اتجاه البيانات عند فصل مسار البيانات كما سنرى .
20. الطرف DEN ، طرف تنشيط مسار البيانات Data bus enable ، حيث يستخدم هذا الطرف لبيان إذا كانت الخطوط AD0-AD15 تحمل إشارة بيانات حقيقة أم لا .
21. الطرف HOLD ، عندما يكون هذا الطرف فعالا (1) فإن المعالج يضع كل مساراته (البيانات ، والعنوانين ، والتحكم) في الحالة الثالثة ، حالة المقاومة العالية ، أو بمعنى آخر فإن المعالج ينفصل عن كل المسارات الخارجية تاركا لها في العادة لتحكم الاتصال المباشر بالذاكرة لاستخدامها في نقل البيانات مباشرة للذاكرة .
22. الطرف HLDA ، خط استجابة للطرف HOLD ويستخدمه المعالج للدلالة على أنه قبل الإشارة HOLD وأنه قد تم الانفصال عن جميع المسارات حيث عند ذلك يقوم المعالج بوضع (1) على هذا الخط .
23. الطرف SS0 في الشريحة 8088 عبارة عن طرف حالة حيث يستخدم مع الأطراف IO/R و DT/M لبيان حالة المعالج في أثناء أي دورة مسار Bus cycle . جدول 2-16 يبين هذه الحالات .
24. الأطراف S0, S1, S2 عبارة عن أطراف حالة تبين حالة المعالج في أثناء أي دورة مسارات وتستخدم هذه الخطوط في الحالة العظمى للمعالج فقط Maximum mode . جدول (3-16) يبين هذه الحالات ، لاحظ التشابه الموجود بينها وبين جدول 2-16 .
25. الأطراف RQ/GT0, RQ/GT1 تستخدم هذه الخطوط لطلب المسارات عن طريق المعالج الحسابي المساعد Math coprocessor الخارجي في الحالة العظمى فقط maximum mode . كل من هذه الخطوط ثنائي الاتجاه حيث عليها يطلب المعالج المساعد المسارات ، وعليها أيضاً يتوجه المعالج ليخبر المعالج

الخارجي بأن المسارات متوافرة . هذه الخطوط ممزوجة مع الإشارات HOLD و HOLD A

<u>IO/M</u>	<u>DT/R</u>	<u>SS0</u>	الوظيفة
0	0	0	حالة اعتراف بالمقاطعة Interrupt acknowledge
0	0	1	قراءة من الذاكرة Memory read
0	1	0	كتابة في الذاكرة Memory write
0	1	1	توقف Halt
1	0	0	تعامل مع شفرة أمر Code access
1	0	1	قراءة من بوابة إدخال I/O read
1	1	0	كتابة في بوابة إخراج I/O write
1	1	1	حالة خمول Remains passive

جدول 2-16

<u>S2</u>	<u>S1</u>	<u>S0</u>	الوظيفة
0	0	0	حالة اعتراف بالمقاطعة Interrupt acknowledge
0	0	1	قراءة من بوابة إدخال I/O read
0	1	0	كتابة في بوابة إخراج I/O write
0	1	1	توقف Halt
1	0	0	تعامل مع شفرة أمر Code access
1	0	1	قراءة من الذاكرة Memory read
1	1	0	كتابة في الذاكرة Memory write
1	1	1	حالة خمول Remains passive

جدول 3-16

26. الطرف LOCK ، يستخدم في الحالة العظمى فقط (طرف 29) ، حينما يكون هذا الطرف فعالا (0) يمنع أي معالج خارجي من الاتصال بالمسارات . بعض الأوامر تستفيد بهذا الخط حتى تضمن اكتمال تنفيذها دون أي تدخل من المعالجات الخارجية .

27. الأطراف QS0-QS1 (الأطراف 25 و 24) عبارة عن خطوط حالة تبين حالة الطابور Queue الموجود في وحدة مواجهة المسارات لبيان إذا كان هذا الطابور فارغ أو ممتلئ . لاحظ أن هذا الطابور يتكون من 6 بait في حالة المعالج 8086 و 4 بait في حالة المعالج 8088 . هذه الخطوط فعالة في الحالة العظمى فقط

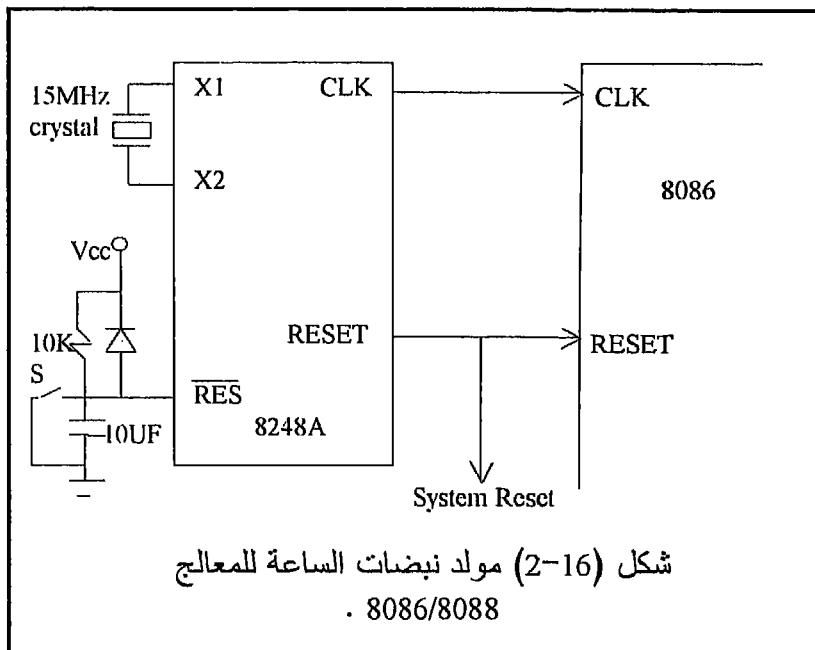
8086 - 1 نبضات الساعة clock للمعالج

كما رأينا سابقاً فإنه لابد من إدخال نبضات ساعة مربعة الشكل سابقة التجهيز إلى الطرف CLK وهو الطرف رقم 19 في الشريحة . تردد هذه النبضات حوالي 5 ميجاهرتز ، و النسبة الدورية لها %33 . هذه النبضات يتم تزامن جميع عمليات المعالج معها . لقد تم إنتاج شريحة بواسطة شركة Intel تعطى هذه النبضات بالمواصفات المطلوبة للمعالج 8086 كما أنها تأخذ الإشارة Reset من المستخدم وتقدمها للمعالج بالمواصفات والتزامن المطلوب ، وكذلك الإشارة READY . هذه الشريحة هي الشريحة 8284A Intel . هذه الشريحة لها طرفان X1 و X2 يوصل عليها بلورة crystal بالتردد المناسب ، كما يوصل عليها المفتاح RESET كما في شكل (16-2) . بالطبع ليس بالضرورة استخدام الشريحة 8284A بالذات ولكن يمكن استخدام أي مولد نبضات بالمواصفات المطلوبة ، ولكن يفضل استخدامها بالذات تجنبًا للكثير من المشاكل وتوفير الكثير من المكونات التي قد تكون مضطراً لاستخدامها .

8086 - 3 عزل مسارات المعالج

إن عملية عزل (فصل) مسارات المعالج 8086 هي نفسها تماماً عملية عزل مسارات المعالج 8085 حيث أن كل من المعالجين يستخدم فكرة المزج الزمني لمسارى البيانات والعنوانين . لذلك فإننا نتحليل القارئ هنا لمراجعة عملية فصل كل من مسار العنوانين والبيانات للشريحة 8085 والتي سبق شرحها . شكل (16-3) يبيّن كيفية فصل مسارى العنوانين والبيانات وبعض خطوط التحكم للشريحة 8086 . نلاحظ من هذا الشكل استخدام ثلاثة شرائح 74373 لمسك إشارة العنوانين في الفترة التي يكون فيها الطرف ALE فعالاً (1) . لاحظ اتصال هذا الطرف بالثلاثة شرائح ، أما الطرف OE لكل شريحة فتم توصيله بالأرضي حتى يكون خرج هذه الشرائح فعالاً دائمًا . بذلك نضمن مسک إشارة خطوط العنوانين A0 - A19 وفصلها عن إشارة البيانات . نلاحظ أيضاً في نفس الشكل

استخدام شريحتين 74245 للحصول على إشارة البيانات . الشريحة 74245 كما نعلم ثنائية الاتجاه ولذلك فإن اتجاه البيانات فيها تم التحكم فيه باستخدام الطرف DT/R الذى تم توصيله على خط التحكم فى الاتجاه DIR فى الشريحة 74245 وذلك نضمن مرور البيانات فى الاتجاه السليم حسب خروجها من المعالج . لابد أيضاً أن تكون الشريحة 74245 فعالة فقط أثناء وجود إشارة بيانات محققة على الأطراف AD15-AD0 لذلك تم استخدام الطرف \overline{DEN} الذى يكون فعالاً (0) كما ذكرنا من قبل عند وجود بيانات محققة على هذه الأطراف . لذلك تم توصيل الطرف \overline{DEN} بالطرف CS للشريحة 74245 .



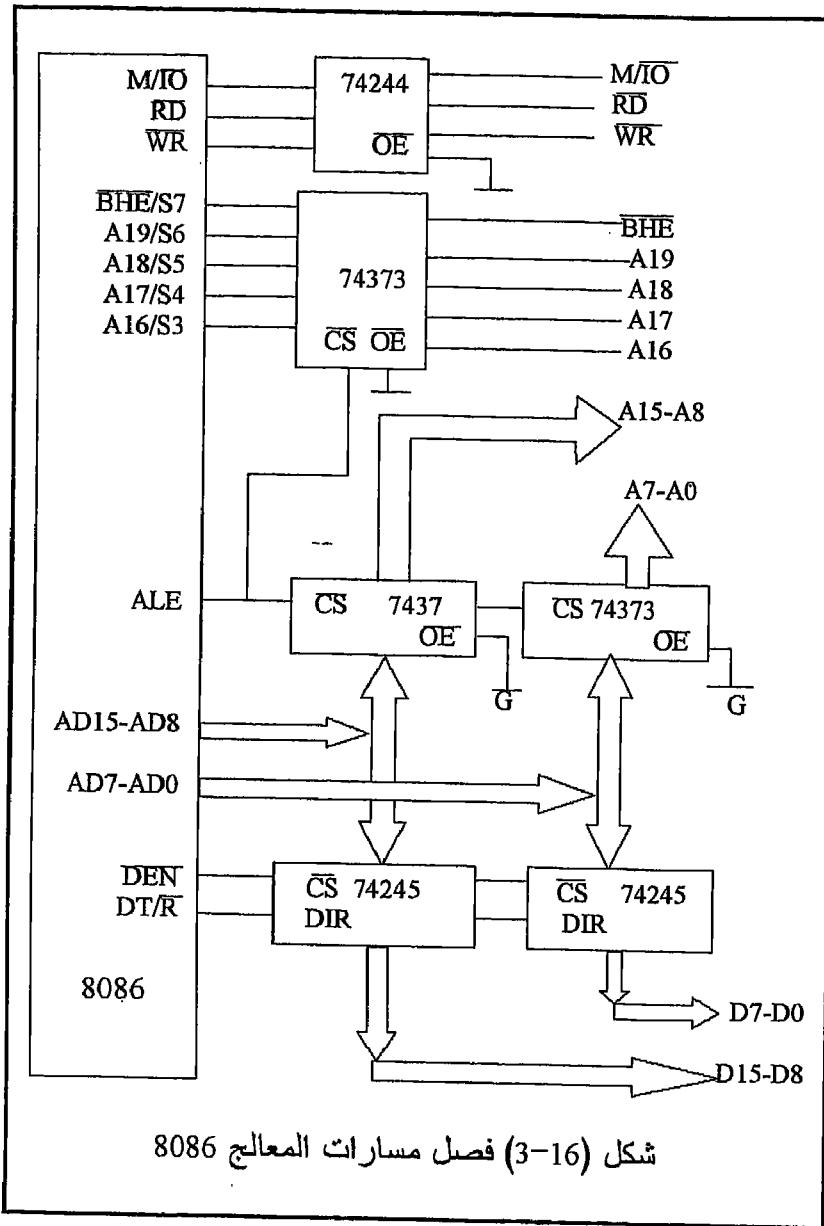
شكل (16-2) مولد نبضات الساعة للمعالج
8086/8088 .

لقد تم استخدام الشريحة 74244 لفصل بعض خطوط التحكم مثل الخطوط $10/\overline{M}$, \overline{RD} , \overline{WR} , وهذه الخطوط غير ممزوجة زمنياً مع أي إشارات أخرى لذلك فعملية الفصل هنا تكون من قبيل عمل حساب الأحمال التي ستوصى على هذه الخطوط .

4-4 مواجهة الشريحة 8086/8088 مع الذاكرة

المعالج 8088 له مسار بيانات 8 بت ومسار عناوين 20 بت ، وعلى ذلك فإنه يستطيع التعامل مع ذاكرة مقدارها 1 ميجابايت . وتم مواجهته مع هذه الذاكرة

بنفس الطريقة التي درسناها مع أي معالج من المعالجات 8 بت التي سبق دراستها ، وننصح هنا بمراجعة الفصل الخاص بمواجهة الذاكرة .



شكل (16-3) فصل مسارات المعالج 8086

كما رأينا فإن شرائح الذاكرة القابلة للقراءة فقط ROM يكون لها خط تحكم في العادة وهو الخط \overline{CS} أو أحياناً يسمى \overline{CE} وهو خط اختيار الشريحة Chip Select أو خط تنشيط الشريحة Chip Enable حيث لابد أن يكون هذا الطرف

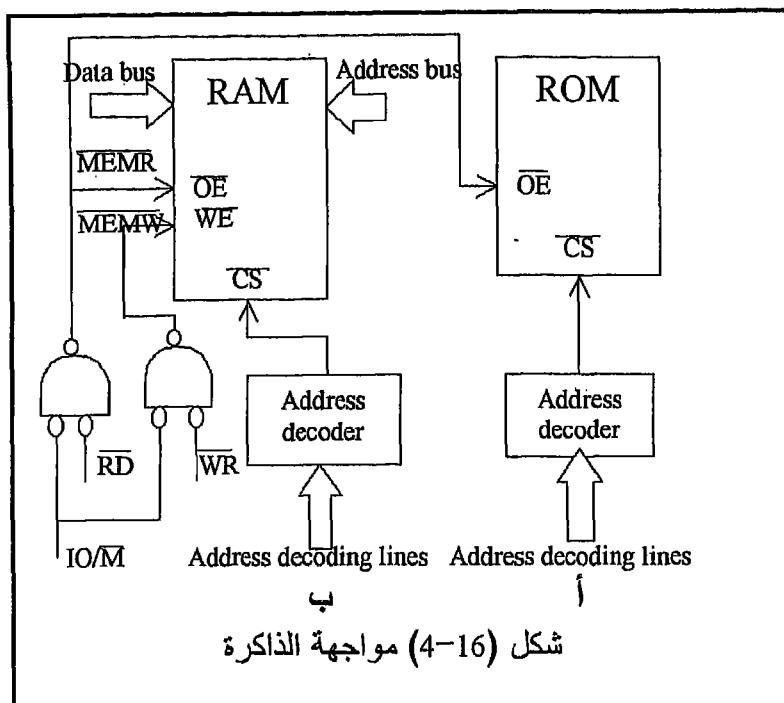
فعالا (0) حتى تعمل هذه الشريحة . هذا الطرف يوصل في العادة بخرج مشفر العنوانين Address Decoder الذي نضمن به أن هذه الشريحة لن تعمل إلا في حالة وجود عنوان يقع ضمن المدى العنوانى لهذه الشريحة والمحدد بمشفر العنوانين . خط التحكم الثاني في شرائح ROM هو الخط \overline{OE} والذي عندما يكون فعالاً يسمح بخروج البيانات المطلوب قراءتها على مسار البيانات لهذه الشريحة وبالتالي يستطيع المعالج قراءتها . هذا الطرف \overline{OE} يوصل في العادة بالإشارة \overline{MEMR} والذي يكون فعالاً عندما يكون كلاً من الخطين \overline{RD} و $\overline{IO/M}$ صفراء .

شكل (16-14) يبين كيفية توصيل شرائح ROM على المعالج . بالنسبة لشرائح ذاكرة القراءة والكتابة RAM يكون لها ثلاثة خطوط تحكم ، أولها يكون الخط \overline{CS} أو \overline{CE} كما ذكرنا سابقاً ، وثانيها يكون الخط \overline{OE} كما هو موجود في شرائح ROM ، وأما الخط الثالث فهو الخط \overline{WE} أو خط تشغيل الكتابة Write Enable والذي يجب أن يكون فعالاً عند التسجيل أو الكتابة في شريحة RAM . هذا الخط يوصل عادة بخط التحكم \overline{MEMW} المكون من الخطين \overline{WE} و $\overline{IO/M}$ القادمان من المعالج . شكل (16-4b) يبين ذلك .

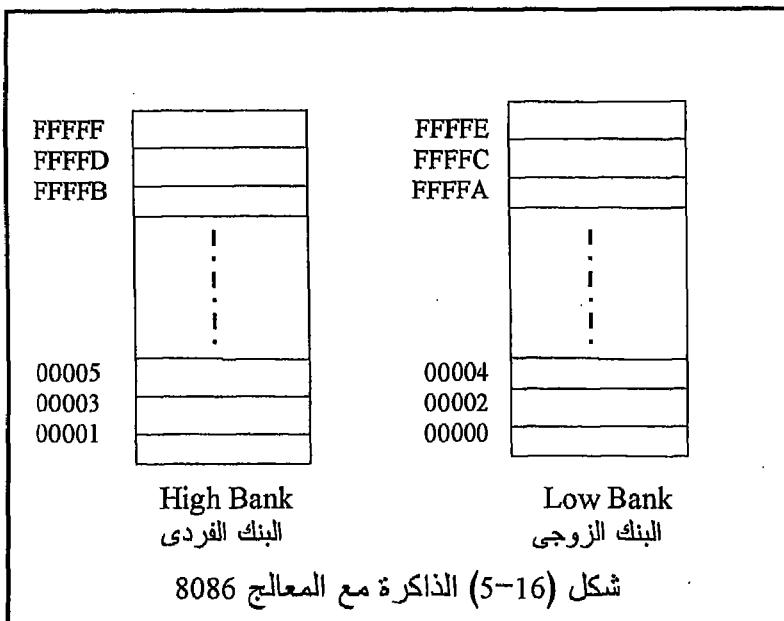
إن مواجهة المعالج 8086 تختلف كثيراً عن مواجهة المعالج 8088 وكل المعالجات السابقة ، حيث أن المعالج 8086 يكون مسار البيانات فيه 16 بت ، ويكون هناك أوامر تتعامل مع هذا المسار على أساس 16 بت وأوامر أخرى تتعامل مع مسار البيانات على أساس 8 بت . لذلك فإن عملية المواجهة هنا يجب أن تأخذ ذلك في الحسبان . مسار العنوانين للمعالج 8086 يتكون من 20 بت ولذلك فإنه سيتعامل مع ذاكرة مقدارها 1 ميجابايت إذا كان سيتعامل على مستوى البایت . أما إذا تعامل على مستوى 16 بت (الكلمة) فإنه سيتعامل مع نصف هذه الكمية ، أي نصف ميجاورد أو 512 كيلوورد . يتم ذلك بالطبع من خلال توصيلة خاصة للذاكرة مع كل من مساري البيانات والعنوانين .

المعالج 8086 ينظر للذاكرة على أنها مقسمة إلى بنيتين . البنك الأول هو مجموعة البيانات ذات العنوانين الزوجية ، والبنك الثاني هو مجموعة البيانات ذات العنوانين الفردية كما في شكل (16-5) . البنك الأول أو الزوجي يسمى البنك الأدنى Lower Half لأنه يحمل نصف المعلومة الأدنى D0-D7 ، ويتم تشغيل هذا النصف باستخدام خط العنوانين A0 الذي يكون صفراء عند التعامل مع أي عنوان زوجي حيث أن أي رقم زوجي تكون أول بิต فيه تساوي صفراء . أما البنك الأعلى من المعلومة D8-D15 فيتم تخزينه في البنك الأعلى من الذاكرة Higher half ويستخدم في ذلك خط التحكم \overline{BHE} الذي يكون فعالاً (0) عند التعامل مع أي بait في البنك العلوي . إذن معنى ذلك أن الخط A0 سيكون صفراء في حالة التعامل مع النصف الأدنى من الذاكرة على أساس 8 بت فقط ، ويكون أيضاً صفراء في حالة التعامل مع الذاكرة على أساس 16 بت . أما الخط

فيكون فعالا (0) في حالة التعامل مع البنك الأعلى من الذاكرة على أساس 8 بت ، أو التعامل مع الذاكرة على أساس 16 بت .



شكل (4-16) مواجهة الذاكرة

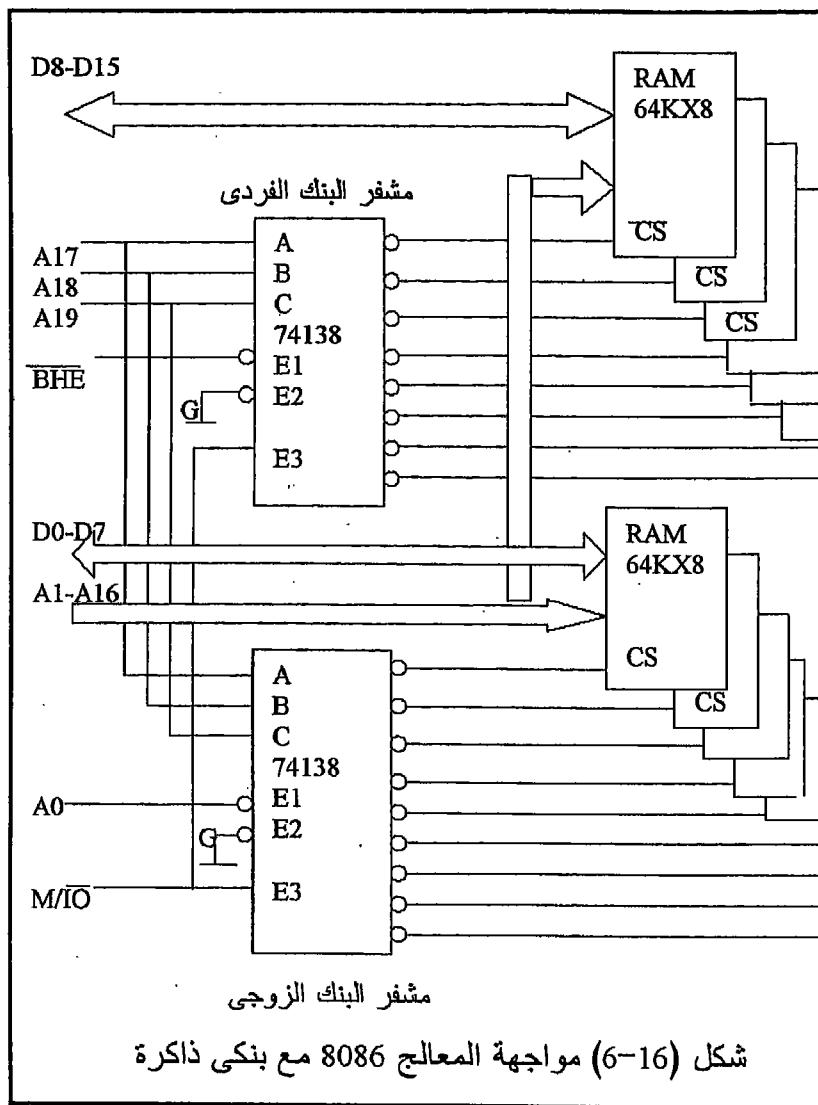


شكل (5-16) الذاكرة مع المعالج 8086

شكل (16-6) يبين كيفية استخدام مشفرين للعناوين للتعامل مع ذاكرة مقدارها 1 ميجابايت مقسمة إلى نصفين زوجي وفردي ، وكل نصف مكون من 8 شرائح كل منها 64 كيلوبايت . لاحظ أن كل من المشفررين تم توصيل خطوط التحكم لهم على خطوط العناوين A17, A18, A19 على التوالي . أما خط التنشيط E3 في كل من المشفررين يتم توصيله على الخط \overline{MIO} والذي يكون واحد عند التعامل مع الذاكرة على العكس من الشريحة 8088 التي يكون فيها هذا الخط صفرًا عند التعامل مع الذاكرة ، ولذلك فإن هذا الخط يرمز له بالرمز $\overline{IO/M}$ في حالة الشريحة 8088 . أما خط التحكم E2 فتم توصيله بالأرضي في الشريحتين ليكون فعالًا دائمًا ، وأما خط التحكم EI في المشفر الأول فمتصل بخط العناوين A0 ليختار النصف الأدنى أو الزوجي من الذاكرة ، والخط \overline{BHE} تم توصيله على الخط \overline{EI} في المشفر الثاني الذي يختار النصف الأعلى أو الفردي من الذاكرة . لذلك فإنه عند التعامل مع الذاكرة على أساس 16 بت فإن كل من A0 و \overline{BHE} يكونان فعالان (0) ويتم التعامل مع نصف الذاكرة في نفس الوقت حيث يتم قراءة أو كتابة البيانات على مسار البيانات بالكامل D0-D15 .

أما عند التعامل مع أحد النصفين الزوجي أو الفردي فإنه إما أن يكون الخط A0 فعالاً (0) أو يكون الخط \overline{BHE} فعالاً (0) وبذلك نضمن التعامل مع أحد النصفين في حالة التعامل على أساس 8 بت .

في شكل (16-6) نلاحظ أن الخط A0 تم توصيله خط تنشيط لمشفر النصف الزوجي من الذاكرة ، ثم بعد ذلك تم توصيل الخط A1 من المعالج بالخط A0 في الذاكرة ، والخط A2 من المعالج بالخط A1 في الذاكرة ، وبذلك نضمن التعامل مع كل بait في الذاكرة في حالة التعامل على أساس 8 بait ؛ أي سيتم استخدام جميع الوحدات ميجابايت من الذاكرة . البعض سيقول لماذا لم تستخدم الخط A0 من المعالج على الخط A0 في الذاكرة وفي نفس الوقت نستخدمه على التوازي لتنشيط المشفر وبباقي خطوط العناوين يتم توصيلها كل مع ما يناظره . حاول أن تخيل هذا التوصيل وتتابع عملية التعامل مع عناوين الذاكرة ابتداءً من العنوان 0000 ثم العنوان 0001 ثم العنوان 0002 وهكذا ستجد أن عملية التخزين ستتم في بait وتترك الأخرى مما يعني أن التعامل سيتم مع نصف كمية الذاكرة فقط ولن يمكن استخدام النصف الآخر منها ، ولذلك يجب تجنب استخدام هذه الطريقة للتوصيل .



16-5 الإدخال والإخراج من وإلي المعالج 8086/8088

لقد سبق شرح عملية الإدخال والإخراج من وإلي المعالجات Z80 و 8085 في فصل سابق ورأينا أن هناك طريقتان للإدخال والإخراج . الطريقة الأولى هي باستخدام الأمرين OUT و IN وسميت هذه الطريقة بطريقة الإدخال والإخراج

المباشر ، كما أن هناك طريقة أخرى للإدخال والإخراج وهي استعمال عنوانين الذكرة في ذلك . وهذا ما أسميه بطريقة خرائط الذكرة للإدخال والإخراج . كلا الطريقتين سنتستخدمهم أيضا مع المعالج 8086 ولذلك ننصح بمراجعة هذا الفصل بالكامل ، لأننا سنبني على ما به من معلومات .

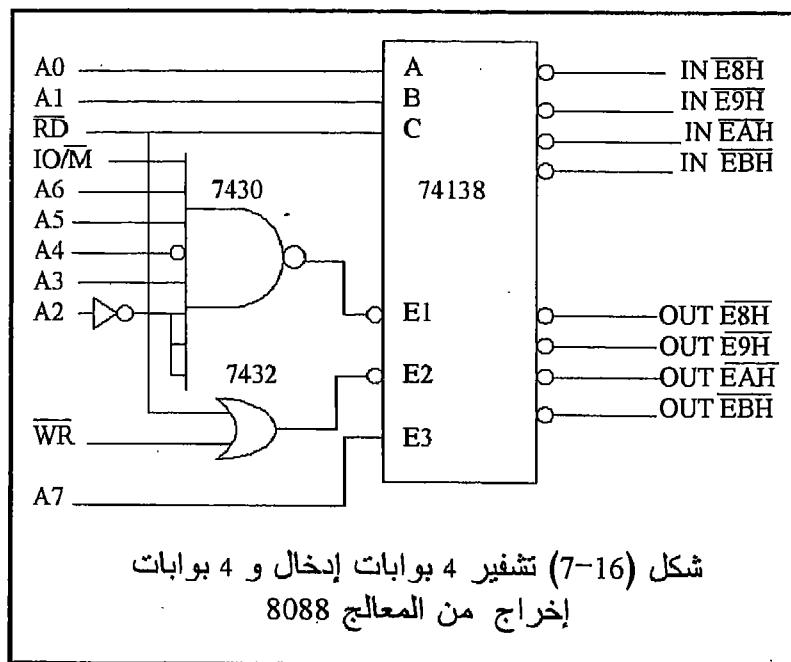
16-5-1 أوامر الإدخال والإخراج للمعالج 8086/8088

جدول 16-4 يبين جميع الحالات الممكنة لأوامر الإدخال والإخراج للمعالج 8088 نلاحظ من هذا الجدول ما يلي :

- عنوان أي بوابة من الممكن أن يكون 8 بت أو 16 بت بحيث عندما يكون العنوان 16 بت فإنه يوضع في المسجل DX . أما إذا كان العنوان 8 بت فإنه يوضع مباشرة في الأمر ، فمثلا الأمر IN AL,DX سوف يقرأ محتويات البوابة التي عنوانها موجودا في السجل DX (16 بت) ويضعها في السجل AL . يمكن أيضا قراءة بوابة مكونة من 16 بت كما في الأمر 0005 AX,0005 I N AX,0005 I حيث سيقرأ البوابة رقم 0005 المكونة من 16 بت ويضعها في المسجل AX (16 بت) . بالطبع فإن الأوامر التي تتعامل مع بوابات 16 بت ستكون محققة فقط مع المعالج 8086 حيث أن له مسار بيانات خارجي من 16 بت كما نعلم .
 - الأمر IN AX,DX سيقرأ محتويات البوابة المكونة من 16 بت والتي يوجد عنوانها في المسجل DX ويوضع هذه المحتويات في المسجل AX . لاحظ أنه طالما أن عنوان البوابة مكون من 16 بت فإن ذلك يعني أنه من الممكن التعامل مع 64ك من بوابات الإدخال والإخراج وهذا كما نرى كم هائل من بوابات الإدخال والإخراج إذا ما قورن بالمعالج 8085 أو المعالج Z80 .
 - جميع أساسيات الإدخال والإخراج التي درسناها في الفصول السابقة مطبقة هنا من حيث أنه لابد من عملية تشفير لعنوان البوابة ، كما أن عملية إدخال البيانات لابد وأن تكون من خلال عازل Buffer ثلاثي المنطق مثل الشريحة 74374 . عملية تشفير البوابات لابد أن تأخذ في الحسبان الطرف M/I/O والذى يحدد متى يتتعامل المعالج مع بوابات إدخال أو إخراج .
- شكل (16-7) يبين مشفر لثمان بوابات ، أربعة منها للإدخال وأربعة للإخراج باستخدام المشفر 74138 . هذا المشفر يستخدم مع المعالج 8088 حيث أنه ثمان بิตات فقط ، ونلاحظ أن عملية التشفير هنا هي نفسها عملية التشفير التي كنا نستخدمها مع المعالجات 8085 أو Z80 . لاحظ استخدام خطوط التحكم IO/M و WR و RD .

الامر	عرض البيانات	تعليق
IN AL,d8	8	قراءة بوابة 8 بت عنوانها 8 بت (d8)
IN AL,DX	8	قراءة بوابة 8 بت عنوانها في (DX)
IN AX,d8	16	قراءة بوابة 16 بت عنوانها 8 بت (d8)
IN AX,DX	16	قراءة بوابة 16 بت عنوانها في (DX)
OUT d8,AL	8	كتابة في بوابة 8 بت عنوانها 8 بت (d8)
OUT DX,AL	8	كتابة في بوابة 8 بت عنوانها في (DX)
OUT d8,AX	16	كتابة في بوابة 16 بت عنوانها 8 بت (d8)
OUT DX,AX	16	كتابة في بوابة 16 بت عنوانها في (DX)

جدول 4-16



بما أن المعالج 8086 يختلف في طريقة تعامله مع الذاكرة عن المعالج 8088 نتيجة الاختلاف في مسار البيانات فإن هذا ينعكس أيضاً على طريقة الإدخال والإخراج للبيانات وبالذات في حالة إدخال أو إخراج بيانات من 8 بت، وستكون المشكلة في هذه الحالة هي هل ستستخدم النصف الأدنى من مسار

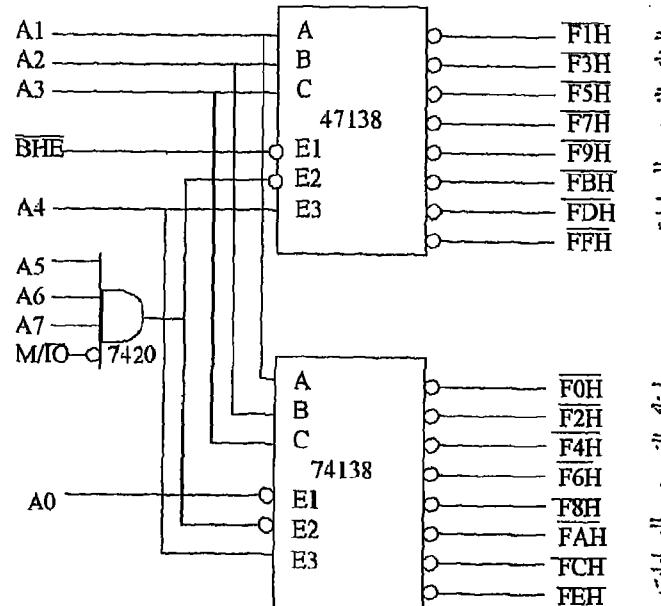
البيانات أم النصف الأعلى ، أم النصفين معا في حالة إدخال أو إخراج بيانات من 16 بت حيث في هذه الحالة لن تكون هناك أي مشكلة ، المشكلة فقط هي في حالة التعامل مع بيانات وبوابات من 8 بت . إن أسهل الطرق لتجنب هذه المشكلة هي التعامل إما مع النصف الأدنى فقط وفي هذه الحالة نجعل الخط A0 فعالا دائما (0) ؛ أو النصف الأعلى فقط وفي هذه الحالة نجعل الخط BHE فعالا (0) . شكل (16-18) يبين ذلك حيث تم استخدام مشفرين 74138 أحدهما يشفّر لثمان بوايات بأرقام زوجية F0, F2, F4, ..., والأخر يشفّر لثمان بوايات فردية F1, F3, F5, شكل (16-18ب) يبين كيفية توصيل المشفّر 74138 لتشفيه ثمان بوايات ذات 16 بت عنوانها هي 80-81 و 82-83 و 84-85 ، لاحظ عدم استخدام الخطين A0 و BHE في عملية التشفير حيث تم الاستغناء عنهما في هذه الحالة . في هذه الحالة مجرد وضع العنوان 80H مثلا سينشط الخرج الأول من المشفّر وبالتالي ينشط نصفى البوابة 16 بت الملحة بهذا الخط معا وفي نفس الوقت .

16-5-2 البوابات القابلة للبرمجة PPI

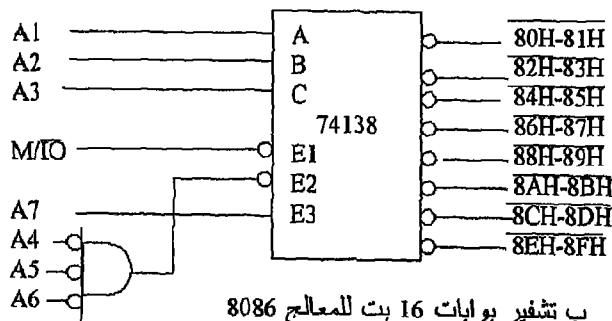
لقد سبق دراسة الشريحة 8255 في فصل سابق و كيفية مواجهتها مع المعالجات ذات 8 بت والأمر لا يختلف كثيرا هنا ؛ لذا نحيل القارئ لمراجعة هذا الفصل . هناك بعض الشرائح الأخرى القابلة للبرمجة والكثير الاستخدام بالذات في مجال الحاسيبات و سنعطي هنا فكرة سريعة عن بعض هذه الشرائح دون الدخول في تفاصيل مواجهة هذه الشرائح . في حالة احتياج القارئ لتفاصيل أكثر عن هذه الشرائح فإننا نحيله إلى الكتالوجات الخاصة بهذه الشرائح والمراجع الموجودة في آخر الكتاب .

16-6 شريحة مواجهة لوحة المفاتيح القابلة للبرمجة 8279

الشريحة 8279 هي شريحة قابلة للبرمجة يمكن بها مواجهة لوحة المفاتيح وكذلك إظهار البيانات التي يتم إدخالها من هذه اللوحة . يمكن توصيل حتى 64 مفتاح على هذه الشريحة ، وتحتوي الشريحة على عازل buffer يمثل طابورا يسمح بتخزين 8 حروف من لوحة المفاتيح ثم استدعاء هذه الحروف على أساس من يصل أولا يخرج أولا عن طريق المعالج . جزء الإظهار يقوم بمسح 16 مكان من أماكن الذاكرة التي تحتوي شفرات البيانات الثمانية المطلوب إظهارها .



أ تشفير بوابات 8 بت للمعالج 8086



ب تشفير بوابات 16 بت للمعالج 8086

شكل (16-8) تشفير البوابات 8 بت و 16 بت للمعالج

7- المؤقت القابل للبرمجة Programmable Interval Timer, PIT 8254

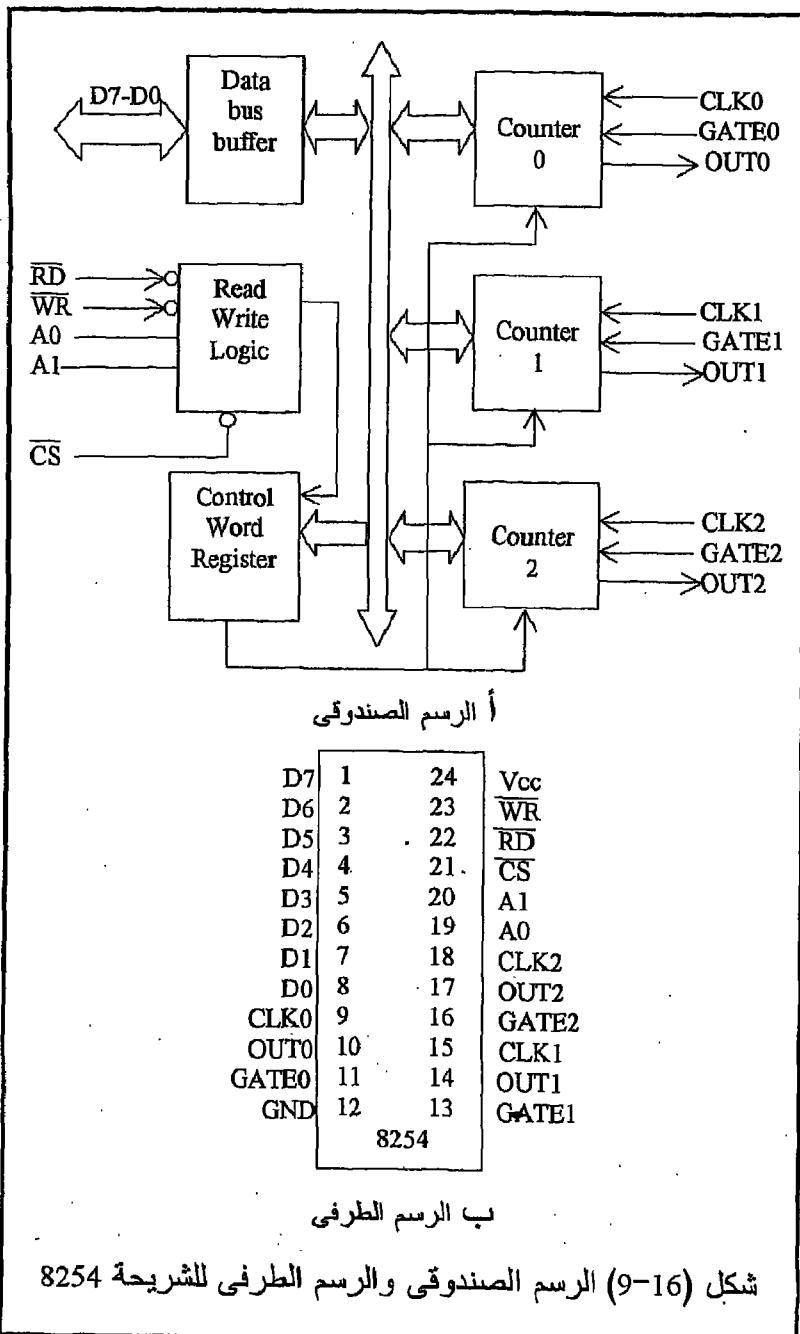
تحتوي هذه الشريحة على ثلاثة عدادات كل منها 16 بت ، وكل منها قابل للبرمجة . كل واحد من هذه العدادات قادر على العد الثنائي أو العد العشري المكون ثنائيا BCD ويتراوح تصل إلى 10 ميجا هيرتز . يمكن استخدام هذه الشريحة في العديد من التطبيقات مثل الساعة الحقيقية Real time clock ، عدادات الأشياء ، التحكم في سرعة موتور واتجاهه مثل موتور الأسطوانة الصلبة أو حتى أي موتور ، والكثير من التطبيقات الأخرى التي يكون الزمن فيها عاملاً مهماً .

إن الحصول على أزمنة تأخير يمكن أن يتم باستخدام بعض أوامر أي لغة من لغات البرمجة ، فمثلاً في لغة الأسمابلي يمكن الحصول على زمن تأخير باستخدام حلقة مغلقة كالتالي :

```
MOV CX,0055H  
XX : NOP  
LOOP XX
```

حيث سيتم تنفيذ هذه الحلقة عدد من المرات مقداره العدد الموجود في المسجل CX . والمعروف أن كل واحد من هذه الأوامر ينفذ في عدد معين من نبضات الساعة الخاصة بالجهاز ، بحيث يمكن معرفة مقدار زمان التأخير بالضبط بمعرفة تردد إشارة نبضات الساعة الخاصة بالجهاز Clock . من المعروف أن نبضات الساعة Clock تختلف من جهاز لأخر ولذلك فإن زمان التأخير الناتج عن الحلقة السابقة لن يكون ثابتاً بأي حال باختلاف جهاز الحاسوب الذي تتفوز عليه هذه الحلقة . إن استخدام شرائح التوقيت مثل الشريحة 8254 يمكن بها الحصول على أزمنة تأخير ثابتة ولن تتوقف على نوع الجهاز الذي تستخدم معه لأنها تعمل على أساس نبضات ساعة ثابتة يتم توصيلها على هذه الشريحة كما سنرى .

شكل (16-19) يبين رسمياً صنديقياً لمحتويات هذه الشريحة وشكل (16-19ب) يبين رسمياً طرفيها لها . الرسم الصنديقي يوضح كيف أن هذه الشريحة مقسمة إلى جزأين أساسيين ، جزءاً يواجه المعالج ، ويكون من ثلاثة أجزاء :
الأول : وهو عبارة عن عازل لمسارات البيانات D7 - D0 ويخرج منه 8 أطراف توصل على مسار البيانات (8 بت القادم من المعالج) .
الثاني : ويحتوي خطوط التحكم في القراءة و الكتابة وهي كالتالي :



- الطرف \overline{RD} الذي يوصل على خط القراءة القادم من المعالج ، حيث عندما يكون هذا الطرف فعالا (0) فإن المعالج يستطيع القراءة من المسجلات الموجودة داخل المؤقت .
- الطرف \overline{WR} الذي يوصل على خط الكتابة القادم من المعالج ، حيث عندما يكون هذا الخط فعالا (0) فإن المعالج يستطيع كتابة أو إرسال بيانات إلى المسجلات الموجودة داخل المؤقت .
- الطرف \overline{CS} ، لكي يمكن للمعالج أن يتعامل مع الشريحة PIT فإن الطرف \overline{CS} وهو خط اختيار الشريحة Chip Select لابد وأن يكون فعالا (0) ، ويكون ذلك بالطبع عن طريق تشفير العنوان الخاص بهذه الشريحة كمارأينا عند التعامل مع الشريحة 8255 .
- الطرفان A0, A1 : هذان الطرفان يوصلان في العادة على خطى العنواين A0, A1 القادمين من المعالج حيث يتم عن طريق هذين الخطين اختيار أحد عدادات الشريحة أو مسجل التحكم داخل الشريحة تبعا للجدول 5-16 .

A1	A0	الوظيفة
0	0	العداد رقم (0)
0	1	العداد رقم (1)
1	0	العداد رقم (2)
1	1	مسجل التحكم

جدول 5-16

الثالث: هو مسجل التحكم Control Register والذي يتم فيه تسجيل كلمة تحكم Control Word من 8 بิตات تمثل اختيار أحد العدادات الثلاثة ليتم التعامل معه حسب حالة تشغيل معينة من خمس حالات تشغيل سنراها بعد قليل .

الجزء الثاني أو الجانب الآخر من الشريحة PIT كما في شكل (16-19) يمثل الجانب المواجه للمستخدم ، وهو يمثل الثالث عدادات الموجودة داخل الشريحة ، حيث لكل عدد منها ثلاثة إشارات أو ثلاثة أطرااف كما يلى :

- الأطرااف CLKX حيث X تمثل رقم العداد (0, 1, 2) ويتم إدخال نبضات الساعة Clock التي سيقوم العداد بعدها على هذه الأطرااف ، ولا بد أن يكون تردد هذه النبضات معروفا جيدا ، ويمكن أن يصل هذا التردد حتى 10 ميجاهرتز .

- الأطراف GATEX وكل طرف منها يمثل طرف تنشيط للعداد المراد التعامل معه . Gate Enable
- الأطراف OUTX وتمثل أطراف خرج للعدادات الثلاثة ، ويمكن برمجة هذه العدادات ليكون الخرج واحداً أو صفراء أو نبضات على حسب حالة التشغيل التي يعمل عليها العداد كما سُرِّيَ .
- آخر طرفان من أطراف الشريحة هما طرفي القدرة GND, Vcc

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD
SC1	SC0	العداد المختار					
0	0	عداد رقم 0					
0	1	عداد رقم 1					
1	0	عداد رقم 2					
1	1	قراءة حالة الشريحة					
		الحالة					
0	0	الحالة 0					
0	0	الحالة 1					
0	1	الحالة 2					
0	1	الحالة 3					
1	0	الحالة 4					
1	0	الحالة 5					
		قراءة/كتابة					
0	0	قراءة/كتابة أمر مسح العدادات					
0	1	قراءة/كتابة النصف الأدنى فقط					
1	0	قراءة/كتابة النصف الأعلى فقط					
1	1	قراءة/كتابة النصف الأدنى ثم الأعلى					
		طريقة العد					
BCD							
0		عداد ثالثي					
1		عداد عشري مكون ثالثياً BCD					

شكل (16-10) كلمة التحكم Control Word للشريحة 8254

8254 برمجة الشريحة

كما رأينا فإن الشريحة 8254 تحتوى على ثلاثة عدادات كل منها 16 بت ، أي يمكن لكل منها أن يعد من 0 إلى FFFFH في حالة العد الثنائي ، أو العد من 0 إلى

9999 في حالة العد العشري . يمكن اختيار أي واحد من العدادات الثلاثة ، وطريقة التعامل معه ، وكذلك حالة التعامل عن طريق شفرة توضع في مسجل التحكم . شكل (16-10) يبين مسجل التحكم ودلالة كل بิต من بิตات هذا المسجل . يحتوى شكل (16-10) أيضا على جداول توضح وظيفة كل مجموعة من مجموعات البتات فى هذا المسجل كما يلى :

- البتات D6 و D7 تمثل شفرات اختيار أحد العدادات Select Counter bits ليتم التعامل معه ، أو قراءة حالة الشريحة . فإذا كان كل من بت 6 و 7 تساوى صفر فإن المقصود في هذه الحالة هو التعامل مع العداد رقم صفر ، أما إذا كانت بت 6 تساوى واحد ، و بت 7 تساوى صفر ، فإن التعامل في هذه الحالة سيكون مع العداد رقم واحد ، وهكذا . أما إذا كان كل من بت 6 و 7 تساوى واحد فإنه في هذه الحالة سيتم قراءة مسجل التحكم .
- البتات D4 و D5 تمثلان كيفية القراءة أو الكتابة من أي واحد من العدادات الذى تم اختياره بالبتات 6 و 7 . كما نعلم فإن كل عداد مكون من 16 بت ، بينما مسار البيانات للشريحة مكون من 8 بت فقط ، لذلك فإنه لابد من تحديد أي بليت (بت) من الـ 16 بت سيتم قراءتها أو الكتابة فيها . فإذا كانت البت 4 تساوى واحد والبت 5 تساوى صفر فإنه في هذه الحالة سيتم التعامل مع البایت الأولى Lower significant byte ، أما إذا كانت البت 4 تساوى صفر والبت 5 تساوى واحد فإنه سيتم التعامل مع البایت الثانية في هذه الحالة Higher significant byte وأخيرا يمكن قراءة البایت الأولى ثم الثانية مباشرة بوضع كل من البت 4 و 5 تساوى واحد . قبل قراءة أي عدد في أي لحظة لابد من مسک Latch قيمة العداد عند هذه اللحظة ووضعها في مسجل القراءة . بذلك نضمن أنه في أثناء قراءة أي بایت فإن البایت الآخر لن يتغير في أثناء القراءة . لذلك فإنه قبل قراءة أي عدد فإنه لابد من مسک محتويات هذا العداد بوضع البتات 4 و 5 كل منها تساوى صفر .
- البتات D1 و D2 و D3 يمكن عن طريقها اختيار الحالة mode التي سيعمل عندها العداد الذى تم اختياره . بهذه الثلاث بิตات يمكن اختيار حالة من ست حالات يمكن لأى عدد أن يعمل عندها كما فى شكل (11-16) .
- البت رقم صفر D0 ويتم عن طريقها جعل العداد الذى يتم اختياره يعد عشري أو ثنائى . فإذا كانت هذه البت تساوى صفرًا فإن العداد المختار سيعد عدًا ثنائياً من 0 حتى FFFFH ، أما إذا كانت هذه البت واحد فإن العداد سيعد عشرياً من صفر حتى 9999 .

16-7-2 حالات تشغيل الشريحة PIT

الحالة 0

شكل (16-11) يبين السنت حالات التي يمكن أن يعمل فيها أي عدد من العدادات الثلاثة الموجودة في الشريحة 8254 . في الحالة 0 وكما هو مبين في شكل (16-11) فإن الخرج OUTX يكون واحد إلى أن يتم تحميل العداد رقم X بالرقم N ويتم تنشيط الخط GATEX حيث عندها ينزل الخرج OUTX إلى الصفر ، ويظل كذلك إلى أن ينتهي العداد X من عد N من نبضات الساعة حيث عند النبضة $N+1$ سيعود الخرج إلى الواحد مرة ثانية . في هذه الحالة لابد أن يكون الخط GATEX نشط دائما .

الحالة 1

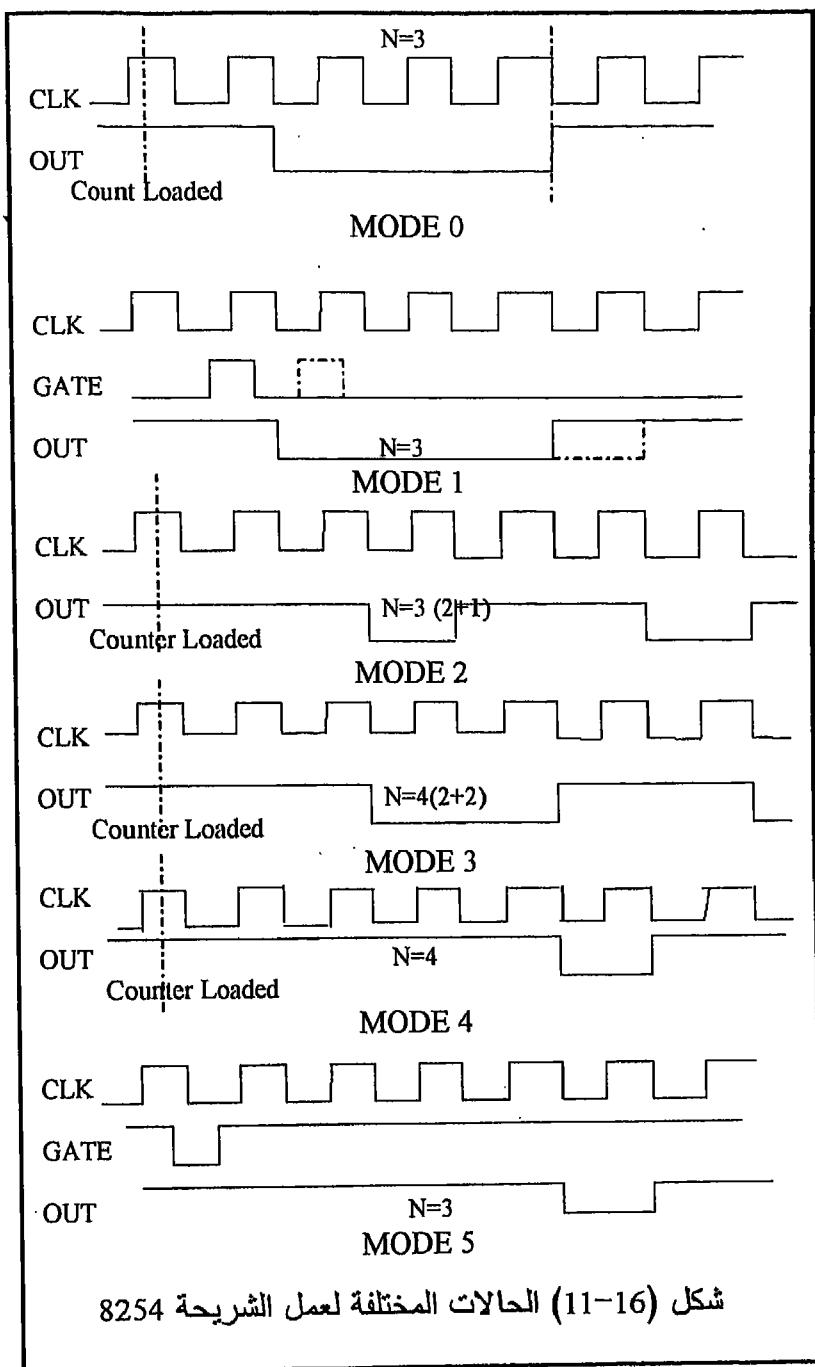
في هذه الحالة سيعمل العداد كمؤقت أحادى الاستقرار يتم تنشيطه من الطرف GATEX حيث بإعطاء نبضة على هذا الخط فإن الخرج سينزل إلى الصفر ويظل كذلك إلى أن يصل العداد إلى القيمة المبرمجة حيث عندها يصعد الخرج إلى الواحد . لاحظ الفرق بين هذه الحالة والحالة السابقة والذي يقع فقط في كيفية تنشيط الطرف GATEX . في هذه الحالة إذا تم إعطاء نبضة تنشيط على الطرف GATEX أثناء نشاط الخرج ، فإن الخرج سيبدأ فترة نشاط جديدة كما في الخطوط المنقطة في شكل (16-11) .

الحالة 2

هذه الحالة يوضحها شكل (16-11) حيث يكون الخرج عديم الاستقرار . في هذه الحالة يكون الخرج واحد طالما أن العداد لم يصل إلى القيمة المبرمجة عليها ، وعندما يصل إلى هذه القيمة فإن الخرج ينزل إلى الصفر لمدة زمن نبضة تزامن واحدة ثم يرجع واحد ، وهكذا يتارجح الخرج بين الواحد والصفر بتعدد وأزمنة تأخير يتم التحكم فيها بالقيمة المخزنة في العداد . لاحظ أن الطرف GATEX في هذه الحالة لابد أن يكون فعالا .

الحالة 3

هنا يكون الخرج أيضا عديم الاستقرار حيث يكون عبارة عن موجة مربعة يتساوى فيها زمن الواحد وزمن الصفر وكل منها له زمن يساوى نصف الزمن الناتج عن القيمة المبرمجة في العداد إذا كانت هذه القيمة زوجية ، أما إذا كان العداد محمل بقيمة فردية فإن زمن الصفر يكون أقل بمقدار زمن نبضة تزامن واحدة عن زمن الواحد .



شكل (16-11) الحالات المختلفة لعمل الشريحة 8254

الحالة 4

بعد مرور عدد من النبضات مساوى لقيمة المحمولة فى العداد ، فإن خرج العداد ينزل إلى الصفر لمدة زمن نبضة واحدة فقط بعدها يعود الخرج واحد كما كان ويظل كذلك إلى أن يتم برمجة العداد مرة أخرى . يمكن إخماد النبضة الناتجة عن طريق جعل الخط GATEX يساوى صفر .

الحالة 5

هذه الحالة تشبه تماماً الحالة 4 سوى أن زمن التأخير يبدأ عند إعطاء نبضة على الخط GATEX حيث بعد هذه النبضة بزمن يتحدد بالقيمة المبرمجة فى العداد ينزل الخرج للصفر لمدة زمن نبضة تزامن واحدة بعدها يرجع الخرج واحد مرة ثانية في انتظار إعطاء نبضة أخرى على الطرف GATEX . لاحظ أن تشغيل زمن التأخير في الحالة 4 يتم برمجياً (فقط يكون الطرف GATEX نشط) بينما في الحالة 5 فإن زمن التأخير يتم تشغيله بالطرف GATEX أى Hardware .

8-16 الاتصالات القابلة للبرمجة

Programmable Communication Interface, PCI 8251

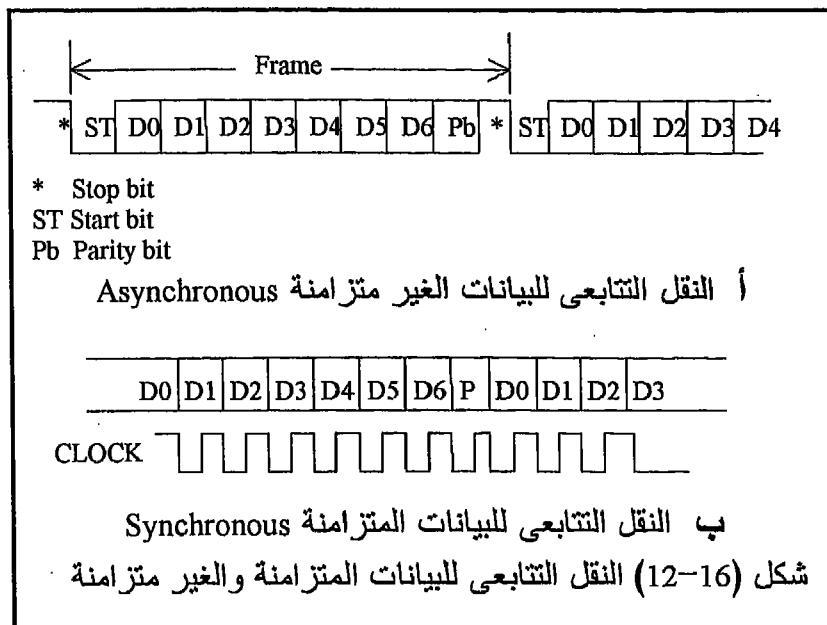
الشريحة 8251 مصممة لتقديم بواجهة نظم الاتصالات التتابعية مع المعالج . كما نعلم فإن عملية نقل البيانات إما أن تكون على التوازي ، أو أن المعلومة ترسل في صورة بايت (8بت) كاملة على خطوط إلى الهدف ، أو ترسل تتابعيناً أي بت بعد بت على خط واحد مثل خط التليفون . في العادة تستخدم الطريقة التتابعية عندما تكون المسافة بين المرسل والمستقبل كبيرة .

الشريحة 8251 عبارة عن مرسل Receiver و مستقبل Transmitter للبيانات الغير متزامنة Asynchronous أو البيانات المتزامنة Synchronous ، ولذلك يرمز لها بالاختصار USART والذى يعني ما يلى :

Universal Synchronous Asynchronous Receiver Transmitter
وتميز الشريحة بمعدلات إرسال baud rate عالية والتى تتمثل في عدد البتات التي يمكن إرسالها في الثانية الواحد .

البيانات الغير متزامنة هي البيانات التي يتم إرسالها واستقبالها دون الحاجة إلى نبضات تزامن Clock . شكل (16-12أ) يبين إطارين من البيانات كل منهما 10 بت ، وكل منها يحتوى على بت البداية Start bit ، وسبع بتات تمثل البيانات

المرسلة Data bits ، ويت للباريتي Parity bit وأخيرا بـت للنهاية Stop bit ، ونلاحظ هنا عدم وجود نبضات تزامن مع هذه البيانات .



إن مهمة الشريحة 8251 هي إضافة بـتات البداية والنهاية والباريتي للبيانات المطلوب إرسالها ، ثم بعد ذلك تقوم بنقل هذه البـتات مع البيانات المطلوب إرسالها على التتابع على خط الإرسال أو قناة الإرسال . عند المستقبل توجد شريحة أخرى من نفس النوع تقوم بالمهامـة العكـسـية حيث تفصل البـتات الإضافـية عن البيانات الأساسية وتحسبـ الـبارـيـتـى هلـ هـىـ سـلـيـمـةـ أمـ لاـ .

أما البيانات المتزامنة فلا تحتوى بـتات إضافـية بـجانـبـ بـتاتـ الـبيـانـاتـ مـثـلـ بـتـاتـ الـبـداـيـةـ وـالـنـهـاـيـةـ وـلـكـنـ جـمـيعـ الـبـتـاتـ تـمـثـلـ بـيـانـاتـ . كلـ بـتـ منـ بـتـاتـ الـبـيـانـاتـ لـابـدـ أنـ تـكـوـنـ مـتـزـامـنـةـ معـ نـبـضـةـ مـنـ نـبـضـاتـ التـزـامـنـ كـمـاـ فـيـ شـكـلـ (16-12ـبـ)ـ ،ـ أـمـاـ بـدـايـةـ إـطـارـ الـبـيـانـاتـ فـتـحـدـدـ بـحـرـفـ تـزـامـنـ .ـ لـنـ نـخـوـضـ فـيـ تـفـاصـيلـ هـذـهـ الشـرـيـحةـ وـطـرـيقـةـ بـرـمـجـتهاـ لـقـلـةـ الـمـتـعـالـمـينـ مـعـهـاـ كـشـريـحةـ مـنـفـصـلـةـ وـلـكـنـ فـيـ الـعـادـةـ يـتـمـ التـعـالـمـ مـعـهـاـ كـأـحـدـ مـكـونـاتـ نـظـامـ اـتصـالـاتـ مـتـكـاملـ .ـ

9-16 الاتصال المباشر مع الذاكرة Direct Memory Access, DMA 8237A

لقد رأينا في طرق التعامل مع الأجهزة الخارجية كيف أنه لكي نخزن معلومة معينة من جهاز خارجي في الذاكرة ، فإننا لابد أن نقرأ المعلومة أولاً عن طريق المعالج ثم نقلها بعد ذلك من المعالج إلى الذاكرة في العنوان المحدد . أي أن المعالج لابد وأن يكون وسيط في عملية نقل المعلومات من وإلى الذاكرة. مع تقدم الحاسوبات وزيادة كمية البيانات التي يتم تداولها بين الأجهزة الخارجية والذاكرة الفعالة أو الأساسية ظهرت هناك فكرة تحرير المعالج من عملية الوساطة هذه بحيث تكون عملية نقل البيانات من الأجهزة المحيطة للذاكرة مباشرة ودون دخول المعالج ك وسيط مما سيسرع من عملية نقل البيانات بدرجة كبيرة ، وهذا ما يطلق عليه الاتصال المباشر بالذاكرة . شكل (13-16) يبين رسمياً توضيحاً لهذه العملية . نلاحظ في هذا الشكل وجود جهاز خارجي يتحكم في هذه العملية وهو عبارة عن الشريحة 8237A والتي تمر من خلالها البيانات من وإلى المعالج دون أن تستقر فيها وإلا فقدنا ميزة السرعة . هذه الشريحة يتحدد دورها في تحديد العنوانين والغرض من التعامل مع الذاكرة هل هو القراءة أم الكتابة . عندما يريد أي واحد من الأجهزة الخارجية مثل الأسطوانة الصلبة أن يتصل مباشرة بالذاكرة ، فإنه يطلب ذلك من المعالج عن طريق تشغيل الخط HOLD الداخلي للمعالج يجعله يساوى واحد . عند ذلك وبعد الانتهاء من تنفيذ الأمر الحالي الذي ينفذه المعالج ، يقوم المعالج بالانفصال عن المسارات الثلاثة (البيانات والعنوانين والتحكم) يجعلها جميعاً في حالة المقاومة العالية أو الحالة المنطقية الثالثة . بعد ذلك يخبر المعالج الجهاز الخارجي بأنه قد انفصل عن المسارات عن طريق تشغيل الخط HLDA يجعله يساوى واحد . عندما يشعر الجهاز الخارجي بذلك يفهم أن جميع المسارات أصبحت تحت سيطرته فيبدأ في إرسال أو استقبال البيانات بمساعدة الشريحة 8237A . يظل المعالج منفصلاً عن المسارات إلى أن يقوم الجهاز الخارجي بإخماد الخط HOLD إلى الصفر مرة أخرى حيث عندها يعود المعالج إلى السيطرة على المسارات مرة أخرى .

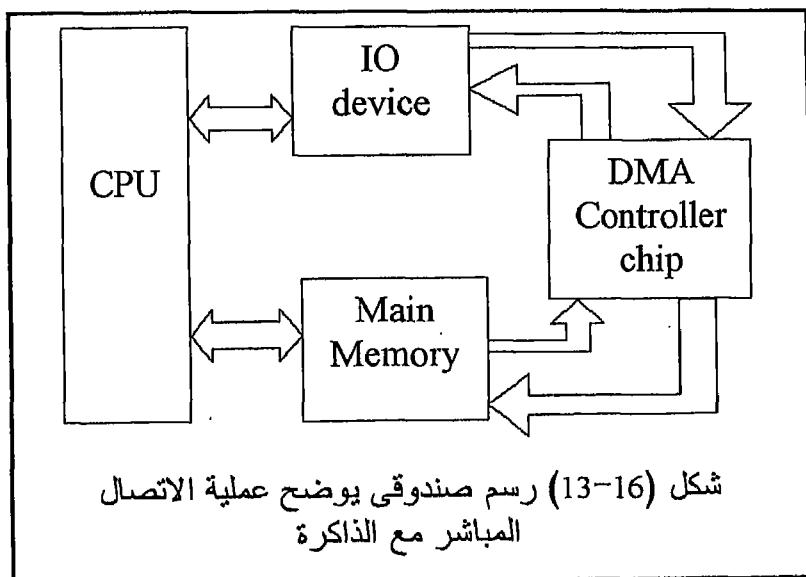
باستخدام الشريحة 8237A يمكن الاتصال المباشر بالذاكرة من خلال 4 قنوات اتصال . سنكتفى بهذا القدر من المعلومات عن هذه الشريحة لندرة استخدامها أيضاً على المستوى الشخصي واستخدامها عادة في الأنظمة المتكاملة مثل أنظمة الحاسوب .

10-16 المواجهة مع المعالجات الحسابية المساعدة Arithmetic Coprocessor 80X87

عائلة المعالجات الحسابية المساعدة Arithmetic coprocessors هي عبارة عن معالجات تقوم بتنفيذ العمليات الحسابية والمقارنات بسرعة تفوق سرعة المعالج العادي حوالي 100 مرة وبالذات الدوال الحسابية مثل دوال حساب المثلثات ودوال الأسس وغيرها . بالإضافة إلى ذلك فإن هذه المعالجات تسهل بدرجة كبيرة التعامل مع البيانات المختلفة مثل الأرقام الصحيحة والحقيقة ذات الدقة المختلفة .

ابتداء من المعالج 8086 حتى المعالج 80386 سنجد أن كل منها له المساعد الحسابي الخاص به والذى يعمل معه ، فمثلاً المعالج 8086 مساعدته الحسابي هو الشريحة 8087 ، والمعالج 80186 مساعدته الحسابي هي المعالج 80187 وهكذا . ابتداء من المعالج 80486 بدأت شركة intell تضع كل معالج ومساعدته الحسابي في نفس الشريحة التكاملية بحيث أصبحت الأنظمة الحاسوبية لا تحتاج إلى المواجهة الخارجية مع المساعد الحسابي .

إننا لن نخوض أيضاً في تفاصيل مواجهة المعالجات المساعدة مع المعالج الأساسي لعدة أسباب منها ندرة استخدامها على المستوى الفردي ، وثانياً أن هذه المعالجات دخلت الآن ضمن مكونات المعالج العادي على نفس الشريحة بحيث أصبحت لا تتنتج بصورة منفصلة .



11-16 تمارين

1. ما هو نوع الإشارة الموجودة على مساري البيانات/العناوين حينما يكون الخط ALE فعالاً؟
 2. ما هو الغرض من خطوط الحالة S3 و S4؟
 3. ما هي الحالة التي يكون فيها المعالج 8086/8088 حينما يكون الطرف RD يساوى صفرًا؟
 4. أشرح الأطراف التالية للمعالج 8086/8088 :
- HOLD
 - HLDA
 - DT/R
 - LOCK
 - TEST
 - READY
5. ما هو الغرض من الطرف BHE؟
 6. لماذا نحتاج في العادة لعملية فصل لمسارات أى معالج؟
 7. كيف نحدد اتجاه الإشارة على مسار البيانات عند استخدام الشريحة 74245 فى عملية عزل المسارات؟
 8. ما هو زمن الاتصال بالذاكرة؟
 9. ما هو الغرض من الطرف DEN؟
 10. ما هو الغرض من الطرف CS والطرف OE فى أى شريحة ذاكرة؟
 11. ارسم المشفر اللازم لعنونة المدى العنوانى DF800H-DFFFFH ؟
 12. ارسم المشفر اللازم لعنونة المدى العنوانى 10000H-1FFFFH باستخدام 8 شرائح EPROM سعة كل منها 8 كيلوبايت .
 13. أضف 8 شرائح RAM أخرى لنظام الذاكرة الموجود في المسألة السابقة ، الشرائح سعة كل منها 2 كيلوبايت . ابدأ المدى العنوانى لهذه الشرائح عند العنوان 20000H .
 14. ما هو الغرض من الطرف A0 غير كونه خط عنونة؟
 15. أشرح كيف نحصل على الخطوط MEMR و MEMW فى المعالج 8086/8088 .

الفصل السابعة عشر

ثم ماذا؟

What else?

1-17 مقدمة

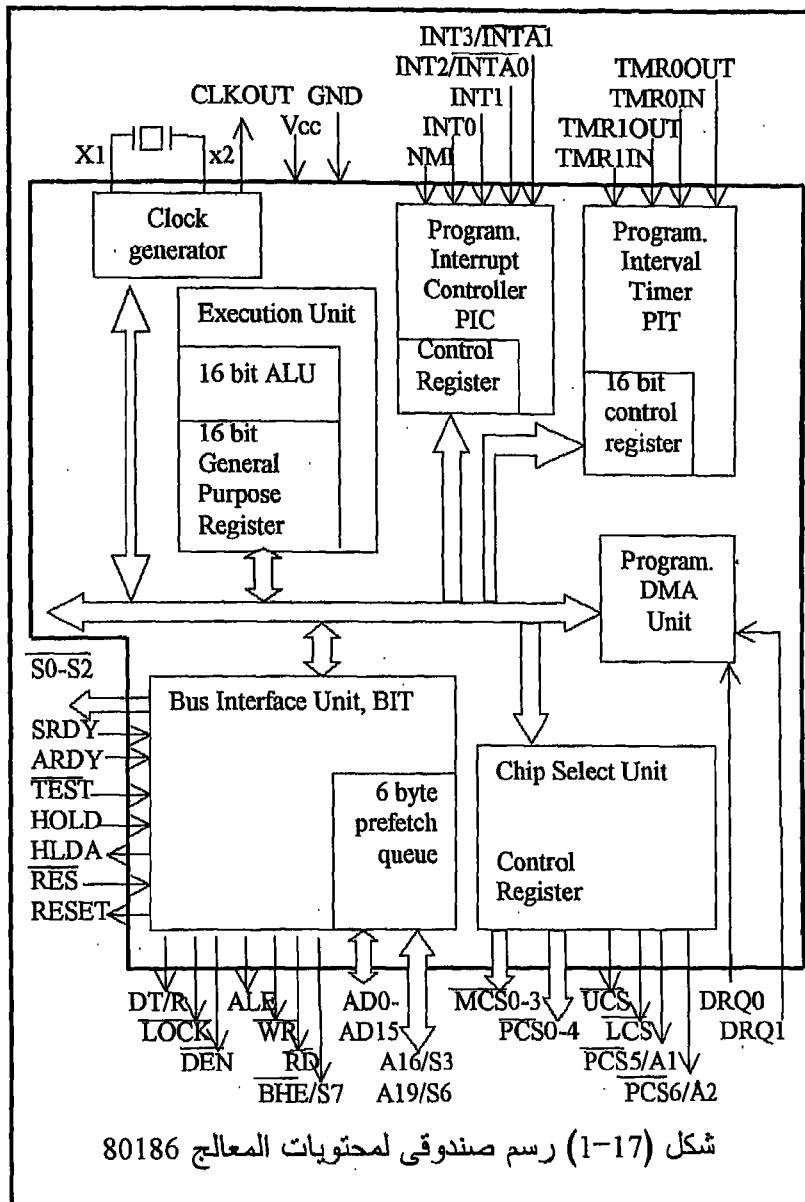
ثم مادا بعد أن درسنا بالتفصيل المعالجين 8085 و Z80 كعينات من المعالجات ذات 8 بت والتي تميز ببساطتها وسهولة برمجتها وسهولة مواجهتها مع الدوائر الخارجية ، ولذلك فإنها غالبا تكون هي المرشحة للاستخدام في بناء دوائر التحكم التي نراها كثيرا في التطبيقات الصناعية والكثير من الأجهزة الحديثة . ثم بعد ذلك درسنا بالتفصيل أيضا المعالج 8086/8088 كأحد المعالجات 16 بت والذي ، كما سنرى ، سيكون هو الأساس لكل المعالجات التالية التي سنراها في هذا الفصل . ولذلك فإننا لن نخوض في تفاصيل هذه المعالجات ولكننا سنكتفى بدراسة الإضافات والفرق التي تمت عليها . سنحاول بقدر الإمكان تغطية جميع المعالجات ابتداء من المعالج 80186 وننتهي بالمعالج بنظام بروتوكول Pentium Pro أحدث المعالجات في الساحة الآن .

2-17 المعالج 80186

شكل (1-17) يبين رسميا صندوقيا لمحاتييات المعالج 80186 . هذا المعالج يشبه إلى حد كبير سابقه المعالج 8086 من حيث مسار البيانات الذي يتكون من 16 بت ومسار العنوانين الذي يتكون من 20 بت . الجديد هنا هو أن الكثير من الشرائح الضرورية التي كان يستعملها المعالج 8086 وكانت توصل معه من الخارج ، تم إدخالها جميعها داخل شريحة المعالج نفسه وذلك لتبسيط دوائر المواجهة مع المعالج 80186 . المعالج 80186 له رفيق آخر وهو المعالج 80188 الذي يشبهه تماما فيما عدا أن مسار البيانات الخارجي يتكون من 8 بت بدلا من 16 بت . مازال كل من المعالجين أيضا يتكون من وحدتين أساسيتين وهما وحدة التنفيذ Execution Unit, EU ووحدة مواجهة المسارات Bus Interface Unit, BIU . شكل (1-17) يوضح البلوكات الأساسية التالية للمعالج 80186 :

1. وحدة نبضات الساعة Clock Generator

هذا المولد يحل محل الشريحة 8284A التي قدمناها في فصل سابق والتي كانت توصل من خارج المعالج لتوفير نبضات الساعة وتوفير عمليات التزامن لكثير من إشارات التحكم مثل الطرف Ready .



شكل (1-17) رسم صندوقى لمحتويات المعالج 80186

هذا البlok يخرج منه الطرفان X1 و X2 اللذان يوصل عليهما باللورة Crystal ذات تردد يساوى ضعف التردد المطلوب للمعالج ، فإذا كان المعالج سيعمل عند تردد 8 ميجا هيرتز مثلاً فإن اللورة يجب أن يكون ترددها 16 ميجا هيرتز . يخرج أيضاً من هذا البlok الطرف CLKOUT الذى يحمل نبضات الساعة الناتجة من داخل المعالج إلى خارجه حتى يمكن استعمالها بأى دائرة خارجية .

2. وحدة منظم المقاطعة القابل للبرمجة

Programmable Interrupt Controller, PIC

يحتوى المعالج 80186 على الشريحة 8259A التى تقوم بتنظيم عمليات المقاطعة حسب أولويات وصولها . هناك خمس مداخل لهذا البلوك وهى خطوط المقاطعة Nonmaskable INT3, INT2, INT1, INT0 وخط المقاطعة الغير قابل للحجب Interrupt NMI .

3. وحدة المؤقتات Timers

يحتوى هذا الجزء على ثلاثة مؤقتات كل منها 16 بت وكلها قابلة للبرمجة مثل الشريحة 8254A والتى تناولناها فيما سبق بالتفصيل . كل هذه المؤقتات يمكنها أن تعمل إما على نبضات الساعة الداخلية الموجودة فى المعالج ، أو مع نبضات خارجية بأى تردد مطلوب .

4. وحدة الاتصال المباشر بالذاكرة

Direct Memory Access, DMA

يحتوى المعالج 80186 على وحدة اتصال مباشر بالذاكرة DMA ذات قناتي اتصال قابلة للبرمجة مشابهة تماماً للشريحة 8237A .

5. وحدة اختيار الشرائح القابلة للبرمجة

Programmable Chip Select Unit, PCS

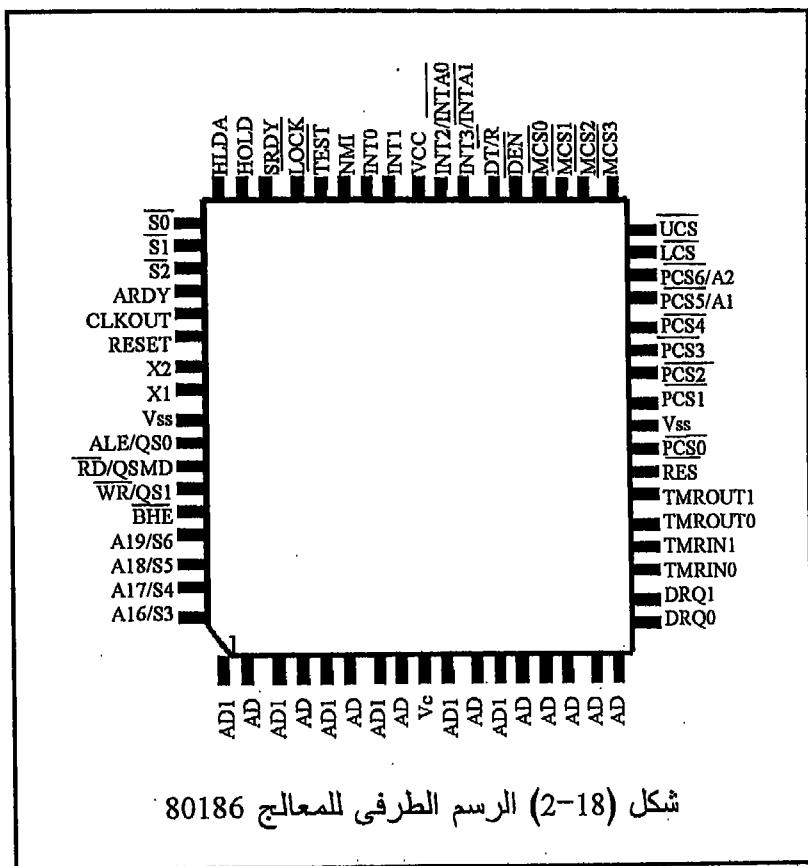
هذه الوحدة عبارة عن مشفر قابل للبرمجة يوفر 6 خطوط لاختيار عنوانين القاعدة base addresses أو عنوانين البداية لمقاطع ذاكرة مختلفة ، كما توفر 7 خطوط لاختيار عنوانين بوابات إدخال أو إخراج . فكر في مدى ما يوفره مثل ذلك من التوصيات الخارجية في حالة تشفير هذه العنوانين خارجياً .

يصدر المعالج 80186 في شريحة مكونة من 68 طرفاً في شكل مربع مختلف عن كل الشرائح السابقة ذات 40 طرفاً . شكل (2-17) يبين رسمياً طرفيـاً لهذا المعالج وفيما يلى سنعرض فكرة مبسطة عن وظيفة كل طرف من هذه الأطراف :

1-2-17 أطراف المعالج 80186

- الطرف Vcc وهو طرف القدرة لهذا المعالج ويساوى 5 فولت وهو الطرف رقم 9 في الشريحة .
- الطرف Vss ويمثل الأرضى الخاص بالشريحة .

- الطرفان X1 و X2 وكما ذكرنا يوصلان على بللورة من الخارج للحصول على نبضات التزامن اللازمة . لاحظ أن تردد النبضات داخل المعالج يكون نصف تردد البللورة .
- الطرف CLKOUT تخرج عليه نبضات التزامن التي تم الحصول عليها حتى يمكن استخدامها بواسطة الأجهزة الخارجية .



شكل (2-18) الرسم الطرفي للمعالج 80186

- الطرف RES وهو طرف إعادة الوضع للمعالج Reset ويجب أن يظل هذا الطرف صفرًا لمدة 50 ميللائية حتى يتم إعادة الوضع . عند تشبيط هذا الطرف يذهب المعالج للعنوان FFFF0H لتنفيذ ما هناك من أوامر .
- الطرفان TMRIN0 و TMRIN1 يتم إدخال نبضات الساعة الخاصة بالمؤقتين 0 و 1 على هذين الطرفين .

- الطرف TEST ، يستخدم هذا الطرف بواسطه الأمر WAIT حيث أنه عندما يكون هذا الطرف فعالا (0) فإنه لن يكون هناك انتظار ، ولكى يتم الانتظار لابد أن يكون هذا الطرف واحد .
- الطرفان TMROUT0 و TMROUT1 وهما طرفا خرج تخرج عليهما إشارة خرج المؤقتين والتي تكون إما فى صورة موجة مربعة أو نبضة واحدة .
- الطرفان DRQ0 و DRQ1 وهما طرفا دخل يتم عليهمما طلب الاتصال المباشر مع الذاكرة DMA من خلال القناتين 0 أو 1 وهما فعالان عندما يكون كل منها 1 .
- الطرف NMI وهو طرف دخل ، تدخل عليه إشارة طلب المقاطعة الغير قابلة للحجب nonmaskable ، وهذا الطرف ينشط مع الحافة الصاعدة للإشارة .
- الأطراف INT0 و INT1 و INT2/INTA0 و INT3/INTA1 ، كلها أطراف دخل تدخل عليها إشارة طلب المقاطعة القابلة للحجب والتي أرقامها 0 و 1 و 2 و 3 وكلها فعالة عندما تكون 1 . هذه الخطوط يمكن برمجتها لتكون 4 خطوط طلب مقاطعة ، أو خطين لطلب المقاطعة وخطين للاعتراف acknowledge بهذه المقاطعة .
- الخطوط A16/S3 و A17/S4 و A18/S5 و A19/S6 ، عبارة عن 4 أطراف تستخدم فكرة المزج الزمني بين إشارة العنوانين A16 إلى A19 وخطوط الحالة S3 إلى S6 . خط الحالة S6 يبين إذا كان المعالج في حالة اتصال مباشر مع الذاكرة حيث أنها يكون هذا الخط 1 ، ويكون صفرا في حالة التشغيل العادي للمعالج . باقى خطوط الحالة تكون أصفارا .
- الخطوط AD0 إلى AD15 ، عبارة عن 16 خط تخرج عليها إشارة العنوانين والبيانات في مزج زمني مثل المعالج 8086 .
- الطرف BHE/S7 طرف خرج يبين إذا كانت الإشارة الموجدة على النصف العلوي من مسار البيانات تمثل بيانات محققة ، هذا الخط ممزوج زمنيا مع الإشارة S7 .
- الطرف ALE/QS0 وهو طرف خرج عبارة عن مزج زمني بين إشارة تشيشط ماسك العنوانين Address Latch Enable, ALE والإشارة QS0 والتي تمثل حالة طابور queue الأوامر في وحدة مواجهة المسارات .
- الطرف WR/QS1 ، خط خرج يبين إذا كان المعالج يكتب بيانات إلى الذاكرة أو وحدة إخراج . هذا الطرف ممزوج زمنيا مع الإشارة QS1 التي تمثل الإشارة الثانية لحالة طابور الأوامر .
- الخط RD/QSMD خط خرج يبين إذا كان المعالج يقرأ من الذاكرة أو من وحدة إدخال . هذا الخط ممزوج زمنيا مع الخط QSMD أو خط بيان حالة الطابور Queue Status Mode .

- الطرف Asynchronous Ready, ARDY طرف دخل للمعالج يخبره إذا كانت الذاكرة أو وحدة الإدخال أو وحدة الإخراج جاهزة Ready . عندما يكون هذا الخط صفر يدخل المعالج في حالة انتظار .
- الطرف Synchronous ready, SRDY هذا الطرف مثلاً مثل الطرف ARDY فيما عدا أنه لا بد وأن يكون متزامن مع نبضات الساعة الخاصة بالنظام . إذا كان هذا الخط صفر يدخل المعالج في حالة انتظار .
- الطرف LOCK خط خرج يبين إذا كان الأمر الذي يتم تنفيذه أمر محظوظ على المعالج المساعد أم لا ، حيث أنه يمكن إضافة بait قبل أي أمر تمنع المعالج المساعد من الحصول على مسارات النظام ، وفي هذه الحالة يكون الطرف LOCK فعالاً ويساوي صفرًا .
- الخطوط S0, S1, S2 أطراف خرج تمثل حالة المعالج أثناء أي عملية نقل للبيانات .
- الطرف HOLD طرف دخل يطلب من المعالج الانفصال عن المسارات لكي تتم عملية اتصال مباشر DMA مع أحد الأجهزة الخارجية .
- الطرف HOLDA طرف خرج يمثل إشارة اعتراف من المعالج بقبول الانفصال عن المسارات .
- الطرف UCS طرف خرج يستخدم كخط اختيار لعناوين الذاكرة في الجزء العلوي من خريطة الذاكرة . يمكن برمجة هذا الطرف لاختيار من 1 كيلو بايت حتى 256 كيلو بايت تنتهي بالعنوان FFFFH .
- الطرف LCS طرف خرج يستخدم كخط اختيار لعناوين ذاكرة في الجزء الأدنى من خريطة الذاكرة . أيضاً يمكن برمجة هذا الخط لاختيار من 1 كيلو بايت حتى 256 كيلو بايت تبدأ بالعنوان 00000H .
- الأطراف MCS0-MCS3 Mid Memory Chip Select خطوط خرج يستخدم كخطوط اختيار لعناوين الذاكرة في أي مكان في الخريطة . يمكن برمجة أي طرف لاختيار من 8 كيلو بايت وحتى 512 كيلو بايت تبدأ عند أي عنوان في الذاكرة .
- الأطراف PCS0-PCS4 خمس خطوط خرج تستخدم لعنونة أجهزة الإخراج والإدخال .
- الطرفان PCS5/A1, PCS6/A1 خطوط خرج تستخدم إما العنونة لأجهزة الإخراج والإدخال مثل PCS0-PCS4 ، أو كخطوط عنونة A0, A1 .
- الطرف DT/R خط خرج يبين اتجاه البيانات على مسار البيانات إذا كانت خارجة أم داخلة للمعالج .
- الطرف DEN يستخدم لتنشيط فوائل مسار البيانات الخارجية حيث يكون هذا الخط فعال (0) في حالة وجود بيانات على مسار البيانات .

80186 برمجة المعالج 2-2

جميع أوامر الشريحة 8086 قابلة للتنفيذ دون أي مشاكل مع المعالج 80186 . يحتوي المعالج 80186 بعض الأوامر الإضافية التي لم تكن موجودة أصلاً مع المعالج 8086 من هذه الأوامر ما يلي :

- ليس هناك أمر في المعالج 8086 يضرب قيمة فورية أو ثابت في محتويات أي مسجل ، فمثلاً الأمر MUL BX, 2300H غير معروف مع المعالج 8086 ولكن مع المعالج 80186 يمكن ضرب أي قيمة فورية في محتويات أي مسجل باستخدام الأمر

IMUL BX, data16

حيث سيضرب الثابت data16 في محتويات المسجل BX ويوضع النتيجة في المسجل BX .

• الأمر 4 SHL BX هذا الأمر يقوم بإزاحة محتويات المسجل BX لليسار بمقدار 4 أماكن أو 4 باتات . في المعالج 8086 كان هناك إمكانية للدوران أو الإزاحة بمقدار بت واحدة فقط .

• هناك بعض الأوامر الإضافية على عمليات الإضافة PUSH والسحب من المكذبة .

بالطبع لابد وأن يكون هناك أوامر إضافية للتعامل مع الشرائح الإضافية والتي أدخلت داخل شريحة المعالج مثل المؤقتات والاتصال المباشر مع الذاكرة والمقطعة .

سنكتفي بذكر هذه الفروق في صورة عامة دون الدخول في تفاصيل وذلك لندرة البرمجة أو الحاجة لهذه الأوامر الإضافية .

3- المـعـالـج 80286

المعالج 80186 لم يستمر كثيراً في السوق ولم يتعد عمر خدمته في أنظمة الحاسوبات سوى عام أو عامين على الأكثر حتى ظهر المعالج 80286 الذي كان بداية نقلة من الحاسوبات XT إلى الحاسوبات AT . المعالج 80286 عبارة أيضاً عن امتداد للمعالج 8086 ويستطيع التعامل مع ذاكرة مقدارها 16 ميجا بايت نتيجة زيادة خطوط العنوانين إلى 24 خطأ بدلاً من 20 خطأ في حالة المعالج 8086 . هذا بالإضافة إلى وحدة جديدة وهي ما يسمى بوحدة إدارة الذاكرة Memory Management Unit, MMU التي بواسطتها يمكن التعامل مع كمية من الذاكرة التخiliية تصل إلى 1 جيجا بايت . هذا بالإضافة إلى أن المعالج 80286

يمكنه التعامل مع أكثر من مستخدم ولذلك يطلق عليه بأنه متعدد المستخدمين . Multi-user أو Multitasking

17-3-1 التركيب الهيكلي للمعالج 80286

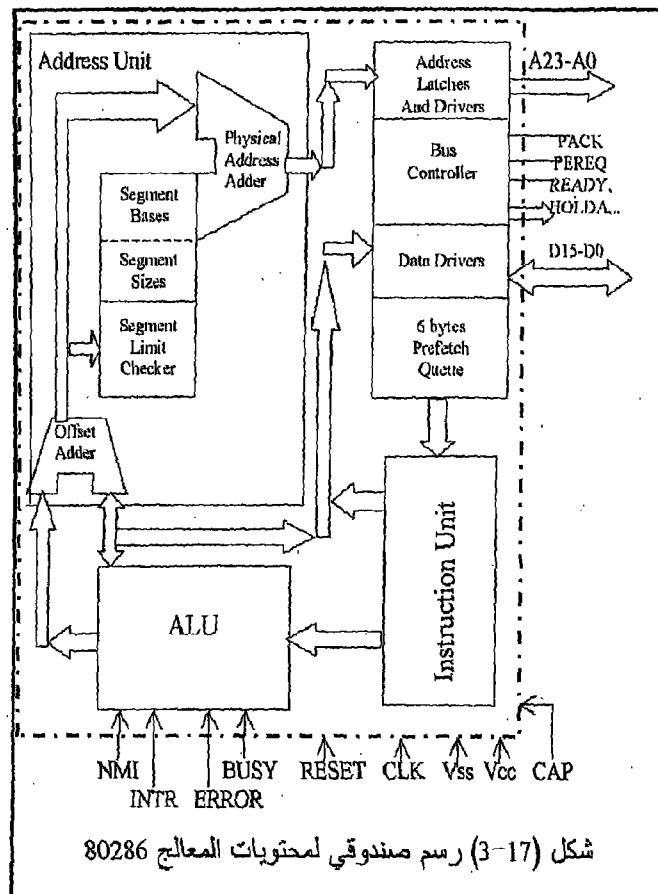
شكل (17-3) يبين رسمياً صندوقياً للمعالج 80286 حيث نلاحظ من هذا الشكل أن المعالج 80286 لا يحتوي شرائط المواجهة التي كانت موجودة في المعالج 80186 ولكن بدلاً من ذلك فإنه يحتوي على وحدة إدارة الذاكرة الجديدة MMU و التي يطلق عليها وحدة العنونة Address Unit في هذا الشكل .

يمكن للمعالج 80286 أن يعمل في واحدة من حالتين ، الحالة الأولى تسمى الحالة الحقيقية real mode وفيها يكون المعالج 80286 مشابهاً تماماً للمعالج 8086 حيث يكون مسار العنوانين في هذه الحالة 20 خطأ فقط مما يسمح بعنونة 1 ميجا بايت ، أما باقي خطوط العنوانين A20-A23 ف تكون أصفاراً في هذه الحالة ، وفي هذه الحالة فإن جميع برمجيات software الشريحة 8086 سوف تعمل مع المعالج 80286 بدون أي تعديل أو أي مشكلة .

الحالة الثانية أو الحالة الجديدة للمعالج 80286 تسمى الحالة المحمية التخiliية protected virtual mode وفي هذه الحالة فإن جميع خطوط مسار العنوانين A0-A23 تستخدم ، مما يتتيح التعامل مع ذاكرة مقدارها 16 ميجا بايت . في هذه الحالة يتم استخدام وحدة إدارة الذاكرة MMU التي تتتيح عنونة حتى 16 كيلو جزء؛ كل جزء مكون من 64 كيلو بايت أي أنها يمكنها عنونة حتى 16 كيلو 64X64 كيلو بايت من الذاكرة التخiliية .

بالإضافة لما تقدم ، تحتوى الشريحة 80286 على بعض المسجلات الإضافية الغير موجودة في الشريحة 8086 المستخدمة في وحدة إدارة الذاكرة . بالطبع فإنه نتيجة إضافة وحدة الذاكرة فلابد أن يكون هناك مجموعة من الأوامر الإضافية التي تستخدم للتحكم في هذه الوحدة ، وهذه المجموعة هي الاختلاف الوحيد في مجموعة الأوامر بين المعالج 80286 والمعالج 8086 .

يستخدم المعالج 80286 فكرة الذاكرة التخiliية virtual memory بحيث يمكن تخصيص جزء من الذاكرة لكل مستخدم user أو كل task . يجب أن نتذكر جيداً أنه عندما يقوم المعالج بتنفيذ عدة برامج لأكثر من مستخدم أو أكثر من هدف على التوازي فإنه في الحقيقة ينفذ جزء من البرنامج الأول الذي يكون ذو أولوية عالية ، وإذا انخفضت أولوية هذا الهدف نتيجة تنفيذ جزء منه ، فإن المعالج يتركه وينفذ في الهدف الثاني أو الثالث ثم يرجع للهدف الأول وهذا ، أي أن عملية التنفيذ تكون موزعة على الأهداف على التتابع ونتيجة السرعة في تنفيذ هذه البرامج يشعر كل مستخدم كما لو كانت كل هذه البرامج تنفذ على التوازي .



شكل (3-17) رسم مسندوقي لمحنيات المعالج 80286

من المشاكل الأساسية الموجودة في المعالج 80286 أنه عندما يدخل في الحالة المحمية التخильية فإنه لا يستطيع الخروج منها والرجوع إلى الحالة الحقيقية real إلا إذا تمت إعادة وضع reset للمعالج ، وهذه بالطبع مشكلة كبيرة لأنها تأخذ وقتاً كبيراً وتفقد كل محتويات الذاكرة . هذه المشكلة تم التغلب عليها في المعالج 80386 .

4-17 المعالج 80386

كانت أول متطلبات هذا المعالج هي تطوير المعالج 80286 بحيث يمكن الرجوع من الحالة المحمية إلى الحالة الحقيقة بسهولة ، وقد كان ذلك حيث يمكن في المعالج 80386 الانتقال من حالة لأخرى باستخدام أمر معين بدلاً من إعادة وضع المعالج ويعتبر هذا إنجازاً كبيراً .

الجديد أيضاً في المعالج 80386 أن مسار البيانات له مكون من 32 بت ، أي أنه يستطيع نقل 4 بait كاملة من أو إلى الذاكرة أو أي جهاز خارجي في رحلة واحدة فقط . كذلك فإن مسار العناوين لهذا المعالج مكون من 32 بت أيضاً مما يتيح له التعامل مع ذاكرة مقدارها 4 جيجابايت . أما إذا دخل المعالج في الحالة المحمية protected mode فإنه في هذه الحالة يتعامل مع 64 تريليون بait (أتريليون بait = 1024 جيجابايت) من الذاكرة التخيلية وذلك باستخدام وحدة إدارة الذاكرة MMU .

17-4-1 التركيب الهيكلي Architecture للمعالج 80386

شكل (17-4) يبين الشكل الخارجي لشريحة هذا المعالج وطريقته الجديدة في ترتيب أطرافه ، حيث يخرج من هذه الشريحة 132 طرفاً مرتبة في صورة شبكة Grid تعرف كل نقطة فيها برقم الصف مقاطعاً مع رقم العمود الذي تقع فيه ، فنقول مثلاً الطرف z13 هو الطرف Vss وهكذا .

المعالج 80386 (اختصار i386) نزل في إصدارين أو صورتين ، الإصدار الأول هو المعالج i386DX وهو الصورة الكاملة لهذا المعالج والتي نحن بصدد دراستها هنا . الإصدار الثاني هو المعالج i386SX الذي يختلف عن الإصدار DX في أن مسار البيانات له يتكون من 16 بت بدلاً من 32 وذلك حتى يتوافق خارجياً مع المعالج 80286 وهذا هو الاختلاف الوحيد بينهما .

17-4-2 تنظيم الذاكرة للمعالج 80386

عندما كان مسار البيانات 8 بت كما في المعالجات 8085 أو Z80 كانت الذاكرة تتنظم في صورة بنك bank واحد ، عرض هذا البنك هو 8 بت (نفس عرض مسار البيانات) . عندما تطور مسار البيانات إلى 16 بت أصبحت الذاكرة تنظم في صورة بنكين كل منها 8 بت بحيث يكون البنك الأول للبيانات الزوجية والثاني للبيانات الفردية ، وكان الخط A0 يستخدم لتشييط البنك الزوجي أو النصف الأدنى في حالة التعامل مع هذا البنك فقط ، والخط BHE يستخدم لتشييط البنك الفردي أو النصف العلوي في حالة التعامل معه فقط ، أما في حالة التعامل

على مستوى 16 بت فإن كل من البنكين يتم تنشيطهما في نفس الوقت من الخطين A0 و \overline{BHE} حتى يمكن إرسال 16 بت (كلمة word) مرة واحدة ، ولقد رأينا ذلك في أثناء دراستنا للمعالج 8086 . شكل (17-5) يبين طريقة تنظيم الذاكرة في المعالجين 8085 و 8086 .

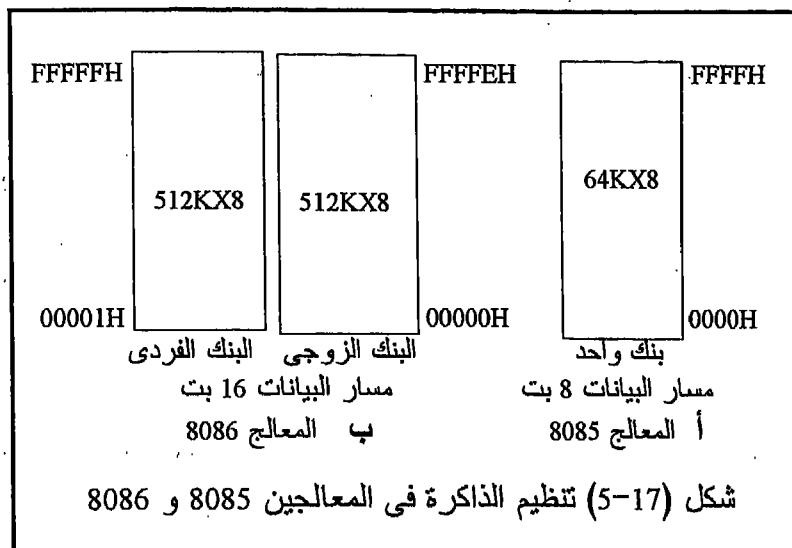
	P	N	M	L	K	J	H	G	F	E	D	C	B	A
1	O	O	O	O	O	O	O	O	O	O	O	O	O	O
	A30	A27	A26	A23	A21	A20	A17	A16	A15	A14	A11	A8	Vss	Vcc
2	O	O	O	O	O	O	Q	O	O	O	O	O	O	O
	Vcc	A31	A29	A24	A22	Vss	A18	Vcc	Vss	A13	A10	A7	A5	Vss
3	O	O	O	O	O	O	O	O	O	O	O	O	O	O
	D30	Vss	Vcc	A26	A25	Vss	A19	Vcc	Vss	A12	A9	A6	A4	A3
4	O	O	O									O	O	O
	D29	Vcc	Vss									A2	NC	NC
5	O	O	O									O	O	O
	D26	D27	D31									Vcc	Vss	Vcc
6	O	O	O									O	O	O
	Vss	D25	D28									NC	NC	Vss
7	O	O	O									O	O	O
	D24	Vcc	Vcc									NC	INTR	Vcc
8	O	O	O									O	O	O
	Vcc	D23	Vss									PEREQ	NMI	ERROR
9	O	O	O									O	O	O
	D22	D21	D20									RESET	BUSY	Vss
10	O	O	O									O	O	O
	D19	D17	Vss									LOCK	W/R	Vcc
11	O	O	O									O	O	O
	D18	D16	D15									Vss	Vss	D/C
12	O	O	O	O	O	O	O	O	O	O	O	O	O	O
	D14	D12	D10	Vcc	D7	Vss	D0	Vcc	CLK	BEQ	Vcc	Vcc	NC	M/I/O
13	O	O	O	O	O	O	O	O	O	O	O	O	O	O
	DI3	DI1	Vcc	D8	D5	Vss	D1	RDY	NC	NC	NA	BE1	BE2	BE3
14	O	O	O	O	O	O	O	O	O	O	O	O	O	O
	Vss	D9	HLDA	D6	D4	D3	D2	Vcc	Vss	ADS	HOLD	BS16	Vss	Vcc

	P	N	M	L	K	J	H	G	F	E	D	C	B	A
80386														

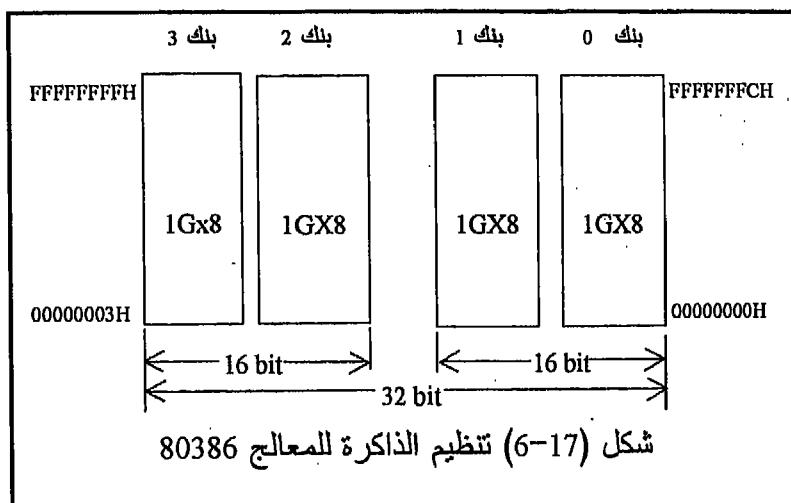
شكل (17-4) الرسم الطرفي للمعالج 80386

مسار البيانات في المعالج i386 مكون من 32 بت ، أي أنه يتعامل مع ذاكرة مقدارها 4 جيجابايت ستقسم كما في شكل (17-6) في صورة 4 بنكبات كل بنك سيكون له خط تنشيط منفصل وهي الخطوط $\overline{BE0}$ إلى $\overline{BE3}$ بحيث أنه عندما يتعامل على مستوى بآيت واحدة فإنه يتم تنشيط البنك المطلوب بخط التنشيط المناسب له ، وعندما يتعامل على مستوى 16 بت فإنه ينشط إما الخطين $\overline{BE0}$ و $\overline{BE1}$ أو

BE1 فى نفس الوقت فى حالة التعامل مع الكلمة الأولى ، أو الخطين BE2 و BE3 فى نفس الوقت فى حالة التعامل مع الكلمة الثانية (العليا) . أما فى حالة التعامل على مستوى 32 بت (4 بait) ففى هذه الحالة تتشط كل الخطوط BE0 إلى BE3 فى نفس الوقت .



شكل (5-17) تنظيم الذاكرة في المعالجين 8085 و 8086



شكل (6-17) تنظيم الذاكرة للمعالج 80386

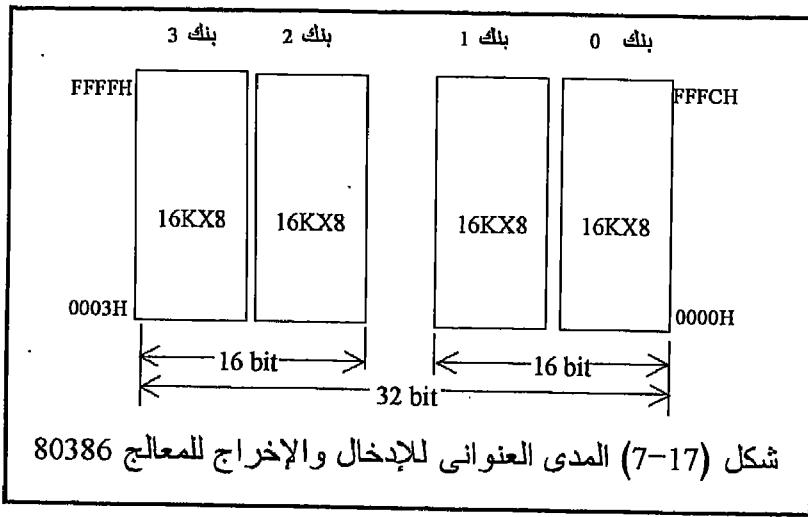
عند إعادة وضع reset المعالج 386 فإنه يذهب إلى العنوان FFFFFFFFOH حيث يبدأ التنفيذ من هناك .

17-4-3 نظام الإدخال والإخراج في المعالج 80386

المعالج 80386 مثل ما سبقه من المعالجات يستطيع التعامل مع عدد من بوابات الإدخال أو الإخراج يصل إلى 64 كيلو ، أى أن المدى العنوانى للإدخال والإخراج هو من صفر إلى FFFFH . الجديد هنا سيكون فى طريقة تنظيم هذه العناوين فى صورة بنكبات نتيجة كون مسار البيانات أصبح 32 بت . نتيجة لذلك سيقسم هذا المدى العنوانى إلى 4 بنكبات كما فى شكل (17-7) حيث ستستخدم خطوط التنشيط البنوك المناسب للتعامل معه سواء كان التعامل فى صورة 8 أو 16 أو 32 بت .

17-4-4 أطراف المعالج 80386

1. الأطراف A0 إلى A31 تمثل مسار العناوين ، وستستخدم لعنونة 4 جيجابايت كما ذكرنا ، الجديد هنا أن خطوط العناوين والبيانات ليست ممزوجة زميلا مع بعضها كما كان الحال فى المعالجات السابقة .



شكل (17-7) المدى العنوانى للإدخال والإخراج للمعالج 80386

2. الأطراف D0 إلى D31 تمثل مسار البيانات .
3. الأطراف BE0 إلى BE3 هى خطوط تنشيط البنوك المختلفة فى الذاكرة والإدخال والإخراج على حسب نظام التعامل 8 أو 16 أو 32 بت .
4. الطرف M/I/O طرف خرج يبين إذا كان العنوان الموجود على مسار العناوين يمثل ذاكرة (حيث يكون هذا الطرف واحد) أم عنوان لبوابة إدخال أو إخراج (حيث يكون هذا الطرف صفر) .

5. الطرف $\overline{W/R}$ طرف خرج يبين إذا كان التعامل الحالى سيكون بغرض القراءة حيث يكون هذا الطرف صفرًا أم الكتابة حيث يكون هذا الطرف واحد . لاحظ أنه في كل المعالجات السابقة كان هناك خطان أحدهما للقراءة \overline{RD} والأخر للكتابه \overline{WR} .
6. الطرف \overline{ADS} طرف خرج يحمل إشارة تبين إذا كانت الإشارة الموجودة على مسار العنوانين تمثل عنوان محقق للذاكرة أو لبوابة إدخال أو إخراج Address . هذا الخط يستخدم في العادة مع الخط $\overline{W/R}$ للحصول على الإشارات \overline{Status} و \overline{MEMR} .
7. الطرف \overline{RESET} ، طرف دخل عندما يكون واحد يسبب إعادة وضع reset للمعالج حيث يذهب المعالج للعنوان FFFFFFFFOH ويببدأ التنفيذ من هناك .
8. الطرف $\overline{CLK2}$ ، طرف دخل يحمل نبضات الساعة Clock للمعالج . تردد هذه النبضات يجب أن يكون ضعف التردد المطلوب للمعالج لأنه يتم قسمة هذا التردد على 2 قبل استخدامه داخل المعالج .
9. الطرف \overline{READY} ، طرف دخل يستخدم لإدخال دورات انتظار على المعالج حينما يكون نشط (0) .
10. الطرف \overline{LOCK} يستخدم لمنع أي جهاز خارجي أو معالج مساعد مثل المساعد الحسابي 387 من الحصول على المسارات .
11. الطرف $\overline{D/C}$ ، طرف خرج يعني Data/Control ويبيّن إذا كانت الإشارة الموجودة على مسار البيانات تمثل بيانات أم إشارة تحكم يخرجها المعالج عند تنفيذ الأمر HALT أو أنه يرسل إشارة اعتراف بالمقاطعة Interrupt Acknowledge .
12. الطرف $\overline{BS16}$ ، طرف دخل يستخدم لتغيير نظام العمل على مسار البيانات بجعله 16 بت بدلاً من 32 بت . حينما يكون هذا الخط صفرًا يتعامل المعالج على أساس أن مسار البيانات 16 بت ، وحينما يكون واحد يعتبر مسار البيانات 32 بت .
13. الطرف \overline{NA} ، ويعني Next Address أو العنوان التالي ، وهو طرف دخل يستخدم لجعل المعالج يخرج العنوان التالي في أثناء تنفيذ الأمر الحالى حيث تستخدم هذه الطريقة لإسراع عملية الاتصال بالذاكرة .
14. الطرفان \overline{HOLD} و \overline{HLDA} مثل نظيريهما في المعالجات السابقة .
15. الطرف \overline{PEREQ} طرف دخل يسمح للمعالج الحسابي 387 بطلب بيانات من المعالج 386 .
16. الطرف \overline{BUSY} ، طرف دخل يستخدم حينما يكون صفرًا لإخبار المعالج بأن المساعد الحسابي 387 مشغول وليس على استعداد لاستقبال أوامر أخرى الآن .

17. الطرف ERROR ، طرف دخل يستخدمه المعالج الحسابي لإخبار المعالج
386 بأن هناك خطأ قد حدث .

18. الطرف INTR ، طرف دخل يستخدم لطلب المقاطعة .

19. الطرف NMI ، طرف دخل يستخدم لطلب المقاطعة الغير ممحوبة .

80386-4-5 مسجلات المعالج

شكل (17-8) يبين التركيب الهيكلي والمسجلات الموجودة داخل المعالج 386 .
نلاحظ من هذا الشكل أن نفس عدد المسجلات مازال موجودا في هذا المعالج
و هذه المسجلات مازالت تؤدي نفس الدور . الجديد هنا هو أن المسجلات في
المعالج 386 تستطيع التعامل مع بيانات مقدارها 8 أو 16 أو 32 بت . بينما
نريد التعامل مع هذه المسجلات على أساس 32 بت فإننا نضع الحرف E
(اختصار لكلمة متعددة (Extended)) أمام المسجل المراد التعامل معه كما في الأمر
التالي :

EAX	AH	AL	AX						
EBX	BH	BL	BX						
ECX	CH	CL	CX						
EDX	DH	DL	DX						
ESP	SP								
EDP	DP								
EDI	DI								
ESI	SI								
المسجلات عامة الأغراض									
<table border="1"> <tbody> <tr><td>CS</td></tr> <tr><td>DS</td></tr> <tr><td>ES</td></tr> <tr><td>SS</td></tr> <tr><td>FS</td></tr> <tr><td>GS</td></tr> </tbody> </table>				CS	DS	ES	SS	FS	GS
CS									
DS									
ES									
SS									
FS									
GS									
مسجلات التجزيء									
EIP		IP							
EFLAGS		FLAGS							
شكل (17-8) مسجلات المعالج 80386									

`MOV EAX,FF340056H`

حيث EAX معناها مسجل التراكم الممتد ، وهكذا باقى المسجلات العامة .
مسجلات التجزء GS, FS, ES, SS, DS, CS تلعب نفس الدور الذى كانت تلعبه مع المعالج 8086 فى الحالة الحقيقية real mode ، وتلعب مع المسجلين GS, FS أدوارا إضافية فى الحالة التخيلية virtual mode .

هناك أيضا المعالج EIP الذى يمثل مؤشر الأوامر الممتد والذى يستطيع التعامل مع 32 بت ، كذلك مسجل الأعلام هنا أصبح ممتدأ أيضا حيث أصبح اسمه . EFLAGS

قبل أن نترك هذا المعالج نؤكد أن جميع أوامر المعالج 8086 مازالت محققة ويمكن استخدامها بالكامل وبدون أى تعديل مع المعالج 386 ، الفرق هو أن المعالج 386 يستطيع التعامل مع بيانات من 8 أو 16 أو 32 بت ، فكل الأوامر التالية صحيحة :

`MOV AL, 55H`

`MOV AX, 5544H`

`MOV EAX, 55443322H`

5-17 الذاكرة المخبأة Cache Memory

مع زيادة نبضات الساعة clock للمعالج (33MHz لمعالج 386) أصبح زمن تنفيذ أي أمر صغيرا جدا بحيث أصبح أقل من زمن الاتصال بالذاكرة مما سيسبب في وجود فترات انتظار عند تنفيذ أي أمر يتعامل مع الذاكرة وبالتالي تقليل سرعة المعالج . زمن الاتصال بالذاكرة هو الزمن اللازم لقراءة أو كتابة وحدة بيانات في الذاكرة ، وهذا الزمن يكون عادة في حدود 50 نانو ثانية بالنسبة للذاكرة الديناميكية . للتغلب على هذه المشكلة تم استخدام أسلوب الذاكرة المخبأة cache ، وهي عبارة عن كمية من الذاكرة السريعة جدا التي تميز بصغر زمن الاتصال بها والتي تصنع خصيصا ، ولذلك فإنها مرتفعة الثمن جدا . كمية هذه الذاكرة تبدأ من 8 كيلوبايت وتصل إلى 512 كيلوبايت وكانت هذه الذاكرة توصل خارج المعالج ، أما الآن فإنها توصل داخل شريحة المعالج نفسه كما سنرى عند عرضنا للمعالجات الحديثة مثل عائلة بنتيوم Pentium .

من المعروف أن التعامل مع بآيات الذاكرة يكون في الغالب من أماكن متتابعة في الذاكرة ، بمعنى أنه عند القراءة أو الكتابة من بآيت معينة فإنه في الغالب يكون التعامل التالي مع الذاكرة من البآيت التالية للبآيت السابقة . لذلك عندما يقرأ المعالج بآيت معينة من الذاكرة فإنه يحضر هذه البآيت وكمية من البآيات

التالية لها ويضعها في الذاكرة المخبأة على أمل أن يكون التعامل التالي مع الذاكرة المخبأة وليس مع الذاكرة الأساسية . ولذلك فإن المعالج عندما يقرأ بآية من الذاكرة فإنه يبحث عن هذه الآية أولاً في الذاكرة المخبأة ، فإن وجدتها فإنه سيقرأها بأقل زمن اتصال ، وإذا لم يجدتها فإنه يحضرها من الذاكرة الأساسية وفي نفس الوقت يضعها أيضاً في الذاكرة المخبأة مع محاولة ملء الذاكرة المخبأة بالبيانات التالية لهذه الآية . عملية ملء الذاكرة المخبأة تتم عادة في أثناء فترات انتظار المعالج . بذلك نضمن أن الآية التي من المحتوى أن يتم قراءتها في المرة القادمة ستكون موجودة في الذاكرة المخبأة . عملية الكتابة في الذاكرة تكون بنفس الطريقة ، فإن كانت المعلومة المراد إرسالها إلى الذاكرة موجودة في الذاكرة المخبأة فإنه يتم نقلها بأقل زمن اتصال ممكن ، وإذا لم تكن موجودة يتم تسجيلها والمعلومات التالية لها في الذاكرة المخبأة أولاً ثم ترسل إلى الذاكرة الأساسية ، بذلك نضمن أن المعلومة التي ستنكتب في الذاكرة في المرة القادمة ستكون موجودة غالباً في الذاكرة المخبأة . أي أن عمليات الكتابة أو القراءة من أو إلى الذاكرة الأساسية تكون من خلال الذاكرة المخبأة ، دون تدخل من المستخدم ، وهذا هو السبب في تسميتها بالذاكرة المخبأة cache لأنها تكون مخبأة عن المستخدم وليس له دخل في التعامل معها أو إدارتها . هذه العملية ثبت أنها تزيد جداً من سرعة التعامل مع الذاكرة اعتماداً على حقيقة أن البيانات التي يتم التعامل معها في أي وقت سيتم التعامل مع المعلومة التالية لها في المرة القادمة .

80486 المعالج 6-17

المعالج 80486 هو معالج عالي التكامل حيث يحتوي بداخله المعالج الحسابي الخاص به 80487 بالإضافة إلى وحدة إدارة الذاكرة وكمية من الذاكرة المخبأة cache memory تبلغ 8 كيلوبايت ، كل ذلك مجمع على نفس شريحة المعالج . لك أن تخيل مدى كثافة المكونات في هذه الشريحة إذا علمت أنها تحتوي على أكثر من مليون ترانزistor . هذا المعالج يستطيع تنفيذ كل أوامر المعالجات السابقة له من عائلته بدون أي تعديل . بالطبع فإنه لابد أن يحتوي على بعض الأوامر الإضافية نتيجة الإضافات التي تضاف عليه . هذا المعالج يستخدم فكرة مجموعة الأوامر المخفضة ، Reduced Instruction Set Computer, RISC والتي ساعدت مع عوامل أخرى في تخفيض الزمن اللازم لتنفيذ الكثير من الأوامر إلى نصف ترازون واحدة . هذا بالإضافة إلى الذاكرة المخبأة cache memory وسرعة نسبات التزامن العالية التي أمكن الوصول إليها في ذلك الوقت والتي بلغت 33 أو 66 ميجاهرتز ، كل ذلك جعل سرعة تنفيذ البرمجيات بهذا المعالج تبلغ أضعاف سرعتها باستخدام المعالج 386 .

يوجد المعالج 486 في صورة شريحة شبكة Grid ذات 168 طرفا . مسار العناوين لهذا المعالج يتكون من 32 بت ، وكذلك مسار البيانات . بالنسبة للتركيب الهيكلي لهذا المعالج فإن مجموعة المسجلات الموجودة فيه هي نفسها مجموعة المسجلات الموجودة في سابقه المعالج 386 . نخلص من ذلك أن المعالج 486 هو نفسه المعالج 386 مضاف إليه المساعد الحسابي 487 وذاكرة مخبأة مقدارها 8 كيلوبايت .

إننا هنا لن ندخل في تفاصيل أخرى عن هذا المعالج ولا المعالجات التالية ، ولكننا سنكتفى فقط بذكر الجديد أو الإضافة التي قدمتها هذه المعالجات وستترك الأمر لمن يريد الاستزادة أن يرجع إلى المراجع الموجودة في نهاية الكتاب أو الكتالوجات الخاصة بالمعالج الذي يريد دراسته بالتفصيل .

7-7 انسيابية الأوامر Instruction Pipelining

ستقدم هنا فكرة جديدة تزيد سرعة تنفيذ الأوامر بدرجة كبيرة جدا تصل إلى خمس مرات على الأقل . تخيل أن أي أمر يحتاج إلى خمس نبضات تزامن حتى يتم تنفيذه ، بحيث تم عملية التنفيذ في خلال الخمس نبضات بالخطوات التالية :

- 1-إحضار الأمر FI
- 2-تشغير الأمر DI
- 3-إحضار المعاملات FO
- 4-تنفيذ الأمر EX
- 5-تخزين النتيجة WR

هذه الخطوات الخمس يمكن تنفيذها بالتتابع على أي أمر ، وفي هذه الحالة فإننا سنحتاج إلى خمس نبضات تزامن لكي تتم عملية إحضار وتنفيذ أي أمر . يمكن إسراع هذه العملية باستخدام فكرة انسياپ الأوامر كما هي موضحة في شكل (7-9) . نلاحظ من هذا الشكل أن كل أمر تم تقسيمه إلى خمس مراحل بحيث عندما يكون المعالج مشغولا في تنفيذ مرحلة معينة لأمر معين فإن الأمر التالي يتم تنفيذه أيضا في نفس الوقت ولكن في مرحلة أخرى من مراحل التنفيذ . فمثلا عندما يكون الأمر رقم I في مرحلة التخزين WR فإن الأمر I+1 يكون في مرحلة التنفيذ EX ، ويكون الأمر I+2 في مرحلة إحضار المعاملات ، والأمر I+3 في مرحلة تشغير الأمر ، والأمر I+4 في مرحلة إحضار الأمر ، وهكذا . أي أنه يكون هناك دائما خمسة أوامر موجودة داخل وحدة التنفيذ كل أمر منها يتم تنفيذ مرحلة معينة منه على حسب موقعة في تتابع الأوامر داخل الوحدة . نلاحظ من هذا الشكل أننا سنحصل من وحدة التنفيذ على أمر وقد تم تنفيذه في

نهاية كل نبضة تزامن (أمر لكل نبضة) . أى أن سرعة تنفيذ الأوامر قد زادت بمقدار خمس مرات ويمكن زيادتها أكثر من ذلك بزيادة عدد مراحل تنفيذ الأوامر . أى أن الأوامر تتاسب في مراحل التنفيذ فيما يشبه الأنبوية أو خط الإنتاج وكل أمر يوجد في مرحلة تنفيذ معينة ، ولذلك سميت هذه الطريقة بالأنسيابية أو Pipelining.

رقم الأمر	نبضات التزامن							
	1	2	3	4	5	6	7	8
I	FI	DI	FO	EX	WR			
I+1		FI	DI	FO	EX	WR		
I+2			FI	DI	FO	EX	WR	
I+3				FI	DI	FO	EX	WR
I+4					FI	DI	FO	EX
I+5						FI	DI	FO

شكل (17-9) الانسيابية Pipelining

من الواضح أنه لكي تستفيد من فكرة الانسيابية فإن جميع الأوامر لابد أن يكون لها نفس الطول أو نفس عدد المراحل ، وكل مرحلة لابد أن تتفذ فى نبضة تزامن واحدة ، فهل هذا محقق فى أوامر المعالجات التى تمت دراستها حتى الآن ؟ بالطبع الإجابة هي لا ، فإن أوامر جميع المعالجات التى درسناها حتى الآن لها أطوال مختلفة وتتفذ فى أعداد مختلفة من نبضات الساعة . وهذا يسوقنا إلى تقسيم المعالجات إلى نوعين من حيث مجموعة أوامر كل منها .

النوع الأول يسمى المعالجات ذات مجموعة الأوامر المركبة ،

Complex Instruction Set Computers, CISC

النوع الثاني يسمى المعالجات ذات مجموعة الأوامر المخفضة ،

Reduced Instruction Set Computers, RISC

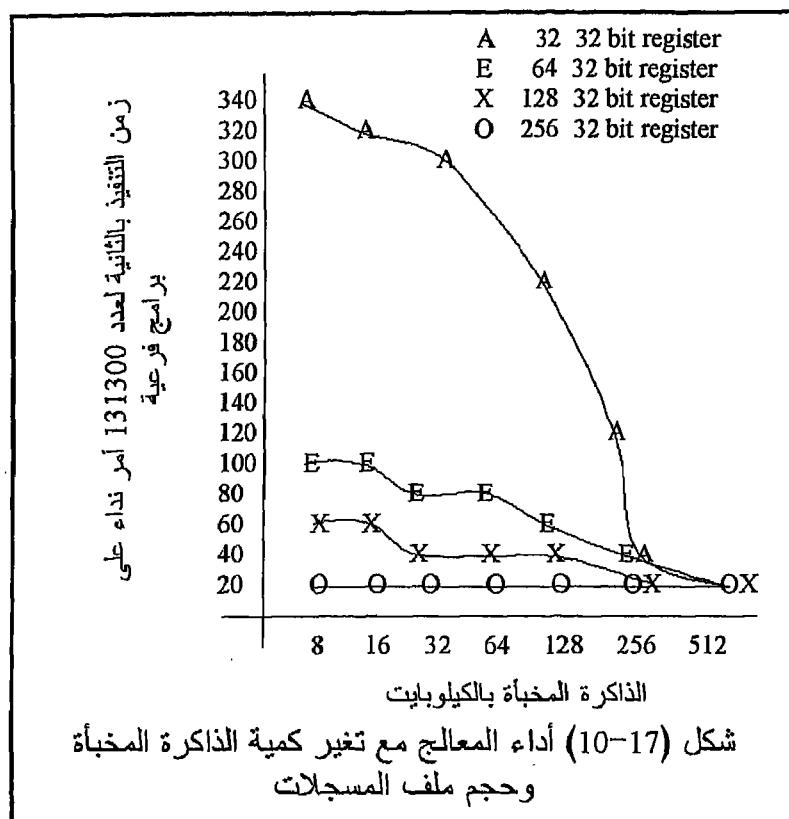
فى النوع الأول من المعالجات ، CISC ، تكون كمية الأوامر كبيرة جدا ، 300 أمر على الأقل ويكون معظم هذه الأوامر أوامر مركبة . ولذلك فإن مشفر الأوامر Instruction decoder يكون معقدا جدا مما يتسبب أن الإشارة تأخذ وقتا كبيرا في تخل دائرة المشفر ، ولذلك فإن وحدة التحكم في هذه المعالجات تكون معقدة . أيضا فإن الكثير من الأوامر المركبة يتم تنفيذها بطريقة البرمجيات الصغيرة Microprograms حيث ينفذ أمر الضرب مثلا بتنفيذ برمجية صغيرة

تتفذ عملية الضرب في صورة مجموعة من أوامر الجمع المتكرر ، وهذا بالطبع يستهلك الكثير من الوقت . كذلك فإنه نتيجة اختلاف أطوال الأوامر في هذا النوع من المعالجات فإنه يكون من الصعب استخدام فكرة الانسيابية Pipelining . نتيجة لذلك ظهر التفكير في النوع الثاني من المعالجات RISC حيث يكون كل شيء هنا مخفض ، عدد الأوامر تم تخفيضه حتى أقل من 128 أمر ، وكذلك طرق عنونة الذاكرة Memory addressing تم تخفيضها حيث أن التعامل مع الذاكرة يستهلك الكثير من الوقت .

مع بساطة عدد الأوامر فإن مشفر الأوامر ، وكذلك وحدة التحكم سيكونان أكثر بساطة مما سيوفر الكثير من وقت التنفيذ . كذلك فإن تخفيض عدد الأوامر سيقتصر على الأوامر ذات الأطوال الواحدة بقدر الإمكان والتي يمكن تنفيذها في نفس عدد المراحل مما سيسهل استخدام طريقة الانسيابية Pipelining في تنفيذ هذه الأوامر . مع تخفيض عدد الأوامر وبساطتها في المعالجات RISC أمكن الاستغناء بدرجة كبيرة على استخدام البرمجيات الصغيرة في تنفيذ الأوامر حيث أمكن استخدام دوائر مبنية Hardware لتنفيذ هذه الأوامر مما وفر الكثير من وقت التنفيذ أيضا . كما رأينا فإن أنظمة RISC تتمتع بالكثير من المميزات مما جعل معظم المعالجات ابتداء من المعالج 80486 تقريبا يأخذ بهذه الفكرة وينفذها ، حيث أصبحت كل الأوامر تقريبا تتفذ في نصفة تزامن واحدة .

في أثناء تنفيذ بعض الأوامر في أي برنامج تنتج هناك بعض النتائج المرحلية التي يحتاجها البرنامج بعد قليل ، ولكن بما أن عدد المسجلات داخل المعالج يكون محدودا فإن المعالج يضطر لإرسال هذه النتائج المرحلية إلى الذاكرة حيث يتم استدعاؤها مرة ثانية عند الحاجة إليها ، وهذا بالطبع يستهلك الكثير من الوقت . هذه المشكلة يمكن التغلب عليها بدرجة كبيرة بزيادة عدد المسجلات ذات الأغراض العامة داخل المعالج بحيث يمكنها أن تستوعب هذه النتائج المرحلية . معظم معالجات RISC تحتوى على عدد كبير من المسجلات تصل إلى 32 مسجلًا وتسمى هذه المجموعة بملف المسجلات Register file . هنا يمكن أن نسأل السؤال التالي : هل يمكن الاستغناء عن الذاكرة المخبأة Cache memory باستخدام ملف مسجلات مع زيادة عدد المسجلات فيه ؟ الإجابة على هذا السؤال هي لا . إن زيادة عدد المسجلات بدرجة كبيرة يجعل من الصعب عنونتها ويكون التعامل مع الذاكرة المخبأة في هذه الحالة أسهل . إن الفاصل بين عدد المسجلات الكبير والصغير غير واضح تماما ولكن ثبت بالتجربة أن 32 أو 64 مسجلًا بالإضافة إلى كمية من الذاكرة المخبأة يكون لها أفضل تأثير على سرعة أداء المعالج . شكل (17-10) [Tabak 1995] يبيّن علاقة بين كمية المسجلات المستخدمة مع كمية الذاكرة المخبأة على أداء المعالج من حيث سرعة تنفيذ مجموعة من أوامر النداء على البرامج الفرعية . من هذا

الشكل نلاحظ كيف أن زيادة عدد المسجلات من 32 إلى 64 مسجلًا كان له تأثيراً كبيراً على زيادة السرعة ، ولكن معدل هذا التحسين كان قليلاً جداً مع زيادة عدد المسجلات عن 64 مسجلًا . ولذلك فإنه ثبت بالتجربة وكما هو واضح من هذا الشكل أن 64 مسجلًا مع 256 كيلوبايت من الذاكرة المخبأة يعطى أحسن أداء للمعالج .



من الأسباب المهمة التي تجعل من الصعب الاستغناء عن المسجلات العامة بالذاكرة المخبأة أن المسجلات العامة يمكن استخدامها بواسطة المبرمج وتكون دائمًا تحت تصرفه ، بينما في حالة الذاكرة المخبأة فإن المبرمج لا يملك أى سيطرة عليها ولا يستطيع التعامل معها على الإطلاق والتحكم في تشغيلها فقط هو وحدة التحكم بالمعالج . لذلك لابد من وجود ملف مسجلات مع كمية من الذاكرة المخبأة للحصول على أحسن أداء للمعالج .

وعلى ذلك يمكن تلخيص مجموعة الخواص المميزة لأى حاسب RISC فيما يلى :

1. جميع الأوامر (أو 80% على الأقل) تنفذ في نبضة تزامن واحدة .

2. جميع الأوامر لها نفس الطول (عدد البايتات لكل أمر) ، وفي الغالب يتكون كل أمر من 4 بايت (32 بت) .
3. عدد مخض من الأوامر لا يتعدي 128 أمر .
4. عدد مخض من طرق التعامل مع الذاكرة Addressing modes لا يزيد عن 4 طرق .
5. كل الأوامر فيما عدا القليل منها يتعامل مع المسجلات فقط .
6. وحدة التحكم تكون مصممة باستخدام الدوائر المبنية Hardwired وليس باستخدام البرمجيات الصغيرة Microprograms .
7. عدد كبير من المسجلات العامة للأغراض (32 مسجل على الأقل) في ملف مسجلات .

- ومن مميزات معالجات RISC ما يلى :
1. البناء باستخدام تكنولوجيا التجميع عالي الكثافة جدا VLSI نتيجة لما يلى :
 - نتيجة تبسيط وحدة التحكم نقل مساحتها بدرجة كبيرة ، ويكتفى أن نعلم أن وحدة التحكم في معالجات CISC تشغّل تقريباً 50% من مساحة الشريحة بينما تشغّل فقط حوالي 10% من مساحة الشريحة في حالة المعالجات RISC .
 - نتيجة صغر مساحة وحدة التحكم على الشريحة يمكن إضافة ملف مسجلات يحتوى عدداً أكبر من المسجلات .
2. زيادة سرعة الحساب نتيجة لما يلى :
 - إمكانية استخدام الانسيابية Pipelining في تنفيذ الأوامر فإن المعالج يكون مشغولاً دائماً مما يزيد من سرعته .
 - تصميم وحدة التحكم باستخدام الدوائر Hardwired بدلاً من البرمجيات الصغيرة Microprograms .
 - وجود ملف المسجلات يقلل التعامل مع الذاكرة .
3. بساطة الأوامر المستخدمة وتخفيف عدددها ، وبساطة تصميم وحدة التحكم بالطبع سينعكس على تكلفة تصميم شريحة المعالج مما سينعكس على سعر المعالج .

من عيوب المعالجات RISC أن تخفيض عدد الأوامر سيضطر مصممو البرامج إلى استخدام عدد أكثر من الأوامر لتنفيذ نفس البرنامج مما سيؤدي إلى كبر حجم البرنامج وكبر حجم الذاكرة التي يشغلها . لذلك فإنه من المتوقع أن برمج معالجات RISC تكون أطول بحوالي 30% من برمج معالجات CISC .

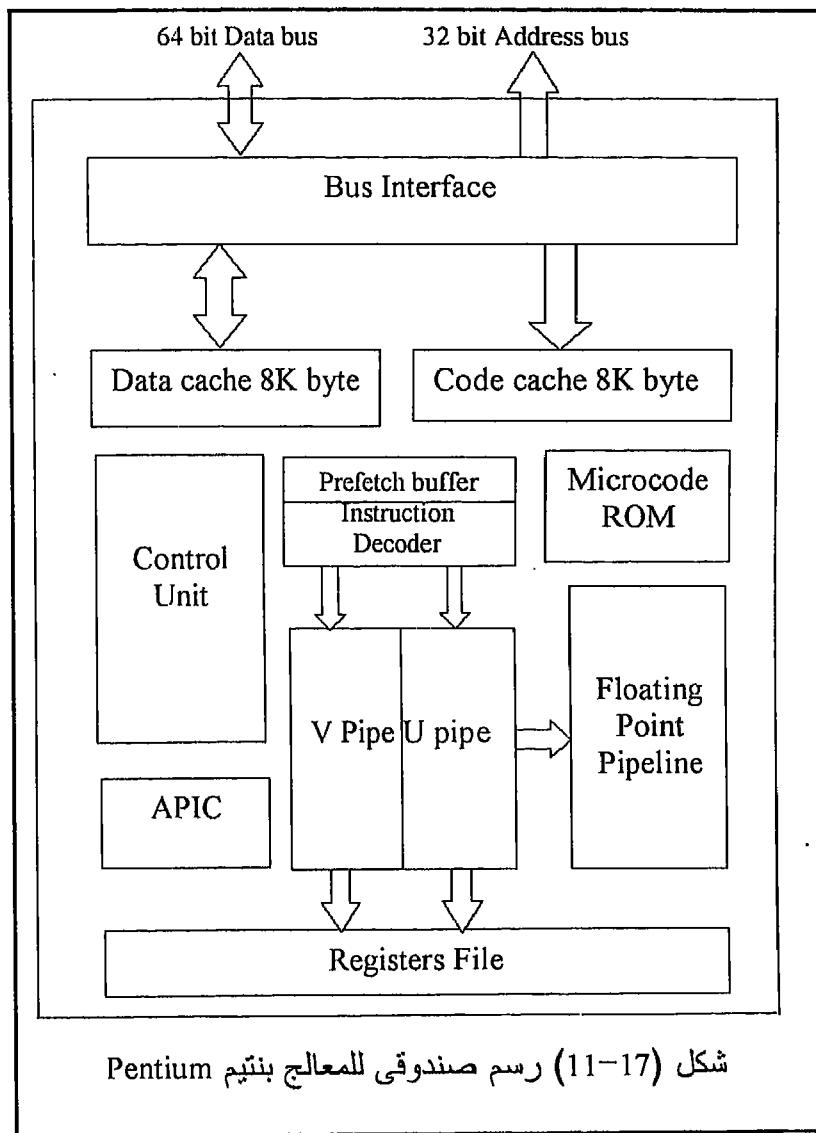
17-8 سلسلة معالجات بنتيوم Pentium Processors

بعد المعالج 80486 وفي بداية التسعينات ، 1993 ، ظهرت سلسلة المعالجات بنتيوم pentium ، وسلطت عليها سلسلة لأنها خضعت لتطورات سريعة جداً ومتلاحقة . هذه المعالجات تستخدم طريقة القوائم المخفضة RISC وأثنان من الانسيابات Pipelines لزيادة سرعة تنفيذ الأوامر حيث وصلت سرعة تنفيذ الأوامر فيه إلى 330 MIPS (Million Instruction Per Second) للمعالج الذي يعمل بنبضات ساعة مقدارها 200 ميجا هرتز بدلاً من 54 MIPS للمعالج i486DX2 الذي يعمل بنبضات ساعة مقدارها 66 ميجا هرتز . من أهم خواص هذا المعالج ما يلى :

1. اثنان من الانسيابات Pipelines واحدة لتنفيذ الأوامر التي تتعامل مع البيانات الصحيحة Integer Pipelines والأخرى لتنفيذ الأوامر التي تتعامل مع البيانات الحقيقة Floating Point Instructions .
2. خاصية توقيع أوامر التفريع مثل الفرز والنداءات على البرامج الفرعية ، والتي يكون لها دخل كبير في إسراع التعامل مع الذاكرة المخبأة .
3. ذاكرة مخبأة خاصة بالتعامل مع البيانات ، وأخرى خاصة بالتعامل مع الأوامر .
4. مسار بيانات خارجي 64 بت .
5. حالة تشغيل جديدة وهي حالة توفير القدرة Power saving mode .

شكل (17-11) يبين رسمياً صندوقياً لهذا المعالج حيث نلاحظ وجود وحدتي الذاكرة المخبأة وحدتي الانسياب المنفصلتين حيث بهذه الطريقة يمكن تنفيذ أمرتين في نفس الوقت على التوازي . أحد إصدارات هذا المعالج يوجد في شريحة ذات 296 طرفاً في الشكل الشبكي وتسحب تياراً مقداره 4 أمبير للمعالج 133 ميجا هرتز . هذا المعالج له 53 طرفاً كلها تمثل القدرة Vcc ، و 53 طرفاً تمثل الأرضي . هذا العدد الكبير من الأطراف الخاصة بالقدرة يستخدم لتخفيف الحرارة المنبعثة . لاحظ ازدواجية وحدة الانسياب وجود وحدة جديدة APIC وهي وحدة تحكم في المقاطعة قابلة للبرمجة Advanced Programmable Interrupt Controller (APIC) . وحدة التحكم الموجودة في المعالج بنتيوم تحكم في دائرة الانسياب حيث في الأحوال العادية يستطيع المعالج تنفيذ أمرتين في نفس الوقت كما ذكرنا . يستطيع المعالج بنتيوم تنفيذ عمليات البيانات الحقيقة Floating Point Instructions .

أسرع من المعالج 486 Point Hardware أدى إلى 10 مرات نتيجة استخدام دوائر مبنية لتنفيذ عمليات الضرب والقسمة بدلاً من البرمجيات الصغيرة . يحتوى المعالج بنطيم على بعض الأوامر المركبة التي تعمل باستخدام البرمجيات الصغيرة وبالذات الأوامر المنقولة من المعالجات السابقة ، ولذلك فإن هذا المعالج متافق تماماً مع كل ما سبقه من معالجات بحيث أن كل أوامر المعالجات السابقة يمكن تنفيذها عليه مع الاستفادة بوحدات الالسياب الموجودة فيه .



شكل (11-17) رسم صندوقى للمعالج بنطيم Pentium

المدى العنوانى للإدخال والإخراج I/O Address Space للمعالج بنتيم يصل إلى 64 كيلوبايت للبوابات ذات 8 بت ، أو 32 كيلوبايت للبوابات ذات 16 بت أو 16 كيلوبايت للبوابات ذات 32 بت حيث يمكن للمعالج التعامل مع كل هذه الأنواع أو مع بعضها .

الذاكرة المخبأة تكون غالباً الثمن كما ذكرنا لأنها تميز بصغر زمن الاتصال بها وتقسم هذه الذاكرة إلى مستويين من حيث اتصالها بالمعالج . فهناك الذاكرة المخبأة ذات المستوى الأول Level 1 والتى يرمز لها بالرمز L1 . هذه الذاكرة يتم بناؤها داخل شريحة المعالج نفسه وتكون كميتها صغيرة حوالى 8 أو 16 كيلوبايت في العادة .

المستوى الثانى من الذاكرة المخبأة Level 2 ويرمز لها بالرمز L2 ، ويتم بناؤها خارج شريحة المعالج وتكون امتداد لذاكرة المستوى الأول . أى أن البيانات تنتقل منها إلى ذاكرة المستوى الأول في حالة القراءة ، ومن ذاكرة المستوى الأول إليها في حالة الكتابة وليس للمستخدم أى سيطرة عليها . كمية هذه الذاكرة تكون كبيرة في العادة حيث تبلغ 512 كيلوبايت .

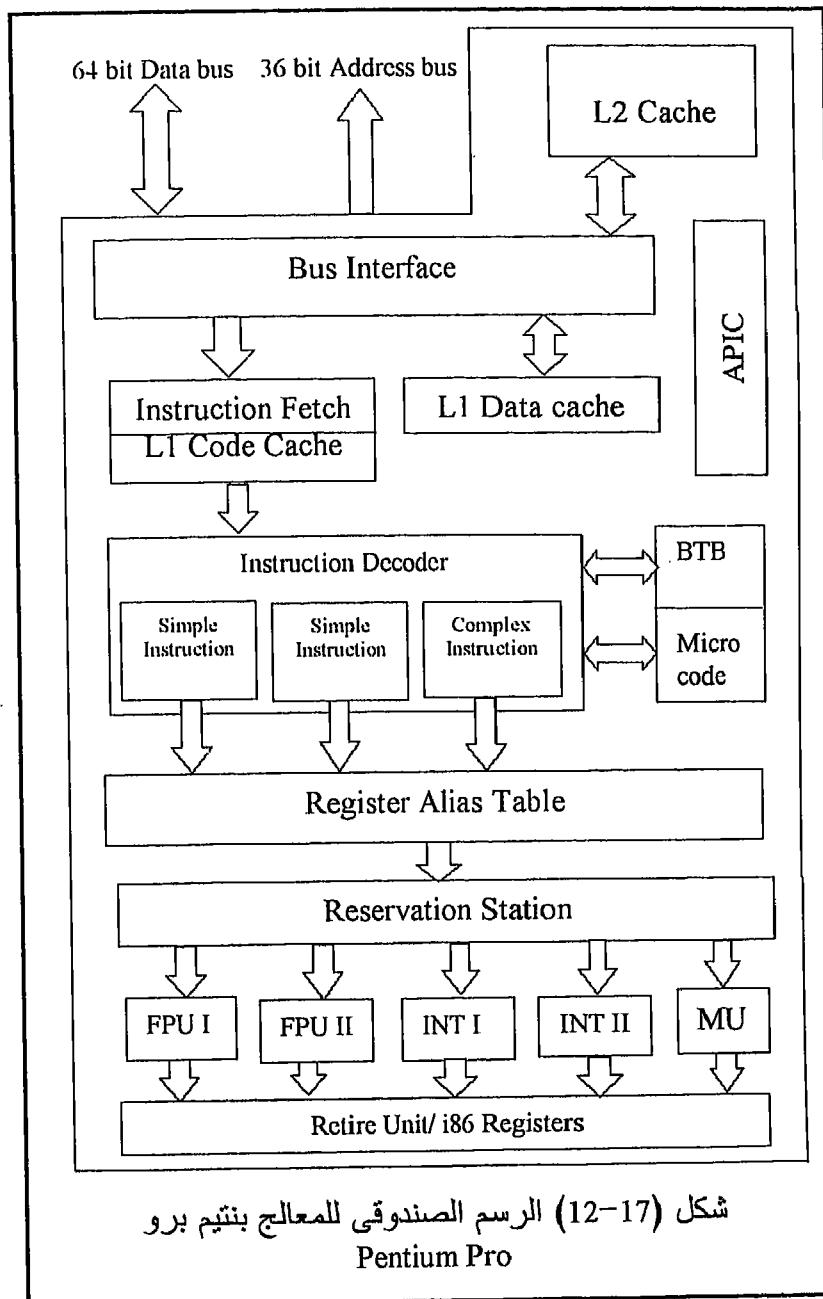
9-17 المعالج بنتيم برو Pentium Pro Processor

لقد كان التطور التالي في عائلة بنتيم هو المعالج بنتيم برو الذي تميز باحتوائه على كل ذاكرة المستوى الثاني المخبأة ، أى أنه يحتوى على 512 كيلوبايت ذاكرة مخبأة من المستوى الأول داخل نفس شريحة المعالج . لكي تتصور مدى كثافة المكونات على شريحة هذا المعالج فإنه يحتوى على خمسة ونصف مليون ترانزistor كلها مجتمعة على شريحة تبلغ مساحتها $7.26 \text{ سم} \times 6.25 \text{ سم}$. هذه الشريحة تستهلك قدرة كهربائية مقدارها 38 وات (12 أمبير تقريباً عند 3.3 فولت) .

يحتوى المعالج بنتيم برو على 3 وحدات انسياپ Pipeline كل منها تتكون من 12 مرحلة ، كما يحتوى على وحدة معالجة البيانات الحقيقية Floating Point Unit, FPU .

لقد صدر المعالج بنتيم برو في شريحة شبکية متداخلة للأرجل Staggered Pin Grid Array, SPGA . شكل (17-12) يبين رسمياً صندوقياً لهذا المعالج . يتضح لنا من هذا الشكل أن مسار البيانات يتكون من 64 بت ، بينما مسار العنوانين يتكون من 36 بت . هناك أيضاً وحدة مواجهة المسارات Bus Interface Unit وتحتى الذاكرة المخبأة والتي تمثل حلقة الوصل بين المسارات الخارجية ووحدة الذاكرة المخبأة

الداخلية ، حيث الوحدة الأولى تمثل وحدة ذاكرة مخبأة للبيانات ، والوحدة الثانية تمثل ذاكرة مخبأة للأوامر وكل منها من المستوى الأول وتبلغ 8 كيلوبايت .



شكل (17-12) الرسم الصندوقى للمعالج بتقىم برو
Pentium Pro

الجديد أيضاً أن كل واحدة من وحدتي الذاكرة المخبأة السابقتين لها مسار بيانات خاص لقراءة البيانات ومسار آخر لكتابتها بحيث يمكن القراءة والكتابة في نفس الوقت من أي واحدة في نفس نبضة التزامن . تمثل وحدة مواجهة المسارات أيضاً حلقة الوصل بين المسارات الخارجية ووحدة الذاكرة المخبأة الثالثة .

تقوم وحدة إحضار الأوامر Instruction Fetch, IF بإحضار الأوامر من الذاكرة المخبأة للأوامر إلى مشفر الأوامر الذي يحتوى على 3 وحدات انسياپ Pipelining تعمل على التوازي ، اثنان منها لتشفيير الأوامر البسيطة ، والثالث لتشفيير الأوامر المركبة CISC والتي تتطلب برمجيات قصيرة لتنفيذها .

يحتوى المعالج بنظام برو على 40 مسجل عاماً تقوم وحدة جدول المسجلات Register Table المبنية في الشكل بالتنسيق بينها وبين المسجلات العامة المعروفة من المعالجات السابقة . يوضح هذا الشكل أيضاً احتواء هذا المعالج على خمس وحدات لتنفيذ الأوامر ، اثنان منها لتنفيذ الأوامر ذات البيانات الحقيقية INTI و FPUII ، واثنان لتنفيذ الأوامر ذات البيانات الصحيحة INTI ووحدة مواجهة الذاكرة Memory Interface Unit, MIU . كل هذه الوحدات تعمل منفصلة وغير معتمدة على بعضها مما يمكنها من تنفيذ خمسة أوامر في نفس الوقت على التوازي وفي نفس نبضة التزامن الواحدة . بالطبع لا يخلو الأمر من بعض الأوامر الشاذة التي تحتاج لمعاملات خاصة وهذه تحل مشاكلها في وحدة العزل Retire unit والتي تحل فيها جميع مشاكل أوامر التفريغ والقفز . آخر وحدة في هذا الشكل هي وحدة المقاطعة المتقدمة القابلة للبرمجة Advanced Programmable Interrupt Controller, PIC والتي تستخدems أساساً لتنشيط عمليات المعالجة المتعددة multiprocessing باستخدام حتى 4 معالجات من هذا النوع دون الاحتياج لأى إضافات .

بذلك تكون قد وقنا على جميع المعالجات بجميع أنواعها ، ونكون قد تأكينا من أن فكرة المعالج الأساسية لم تتغير ابتداءً من المعالجات 8 بت وانتهاءً بالمعالجات 64 بت .

10-17 تمارين

1. ما هو مقدار الذاكرة التي يمكن أن يتعامل معها كل من المعالجات التالية :

- المعالج 8086
- المعالج 80186
- المعالج 80286
- المعالج 80386
- المعالج 80486

- المعالج بنتم
 - المعالج بنتم برو
2. كم عدد بذات مسار البيانات في كل المعالجات السابقة ؟
 3. اشرح نظام الذاكرة في كل واحد من المعالجات السابقة وكيفية تشغيل البنوك المختلفة في كل حالة ؟
 4. اشرح نظام الإدخال والإخراج في كل من المعالجات السابقة وكيفية تشغيل البنوك المختلفة في كل حالة أيضا ؟
 5. ما هي وظيفة الطرف BS16 في المعالج 80386 ؟
 6. ما هي الذاكرة المخبأة cache memory ؟ وما هو أول معالج بدأ في استخدامها ؟ وما مقدارها في كل معالج من المعالجات التي استخدمتها ؟
 7. اشرح المقصود بملف المسجلات Register file ؟ وما هو أول معالج بدأ في استخدامه ؟ وما هو عدد المسجلات في كل معالج من المعالجات التي استخدمت هذه الفكرة ؟
 8. ما هو الفرق بين الذاكرة المخبأة وملف المسجلات ؟ وهل يمكن الاستغناء عن أي منها على حساب الآخر ؟
 9. اشرح فكرة الانسياب Pipelining ؟ وما هي المعالجات التي تستخدم هذه الفكرة ؟
 10. ما هو المقصود بالحاسبات ذات القائمة المخفضة RISC ؟ وعلى أي شيء تم التخفيض ؟
 11. هل هناك علاقة بين فكرة تخفيض الأوامر RISC والانسياب ؟
 12. اذكر خصائص ومميزات وعيوب الحاسبات RISC ؟
 13. ما هو الفرق بين المعالجين بنتم وبنتم برو ؟
 14. إذا طلب منك تصميم دائرة تحكم في إشارة مرور في تقاطع معين ، فرأى المعالجات التي درستها تختار ؟
 15. ما رأيك في الاستغناء عن دراسة المعالجات 8 بت والاكتفاء بدراسة آخر الأجيال منها وهو المعالج بنتم برو مثلا ؟

الملحق الأول ... 1

الحساب الثنائي Binary Arithmatic

مقدمة

لقد رأينا في الفصل السابع كيفية إجراء الجمع الثنائي والطرح الثنائي باختصار شديد وقد كان التركيز الكلى على الدوائر التي تفى بهذه الأغراض ، ولكن تبقى حتى الآن بعض الأسئلة التي مازالت تحتاج إلىيضاح ومنها كيف يتعامل المعالج مع الأرقام السالبة ؟ وكيف يميز بين رقم سالب وآخر موجب ؟ وأنا كمبرمج كيف أعرف إذا كانت النتيجة سالبة أو موجبة ؟ وماذا عن عمليات الضرب والقسمة ؟ كل هذه الأسئلة سنجيب عنها في هذا الملحق إن شاء الله .

عمليات الجمع والطرح

انظر إلى عمليات الجمع الثنائي التالية :

BCD	عشرى	BCD	عشرى
0010 0101	25	0101 0101	55
0011 0001 +	31 +	0100 0100 +	44 +
0101 0110	56	1001 1001	99

نلاحظ في هذين المثالين التطابق التام في نتيجة الجمع في كلا النظامين العشري والعشرى المكود ثنائيا BCD . الآن انظر إلى عمليتى الجمع التاليتين :

DCB	عشرى	BCD	عشرى
0010 1001	29	0101 0101	55
0010 0010 +	22 +	0011 1000 +	38 +
0100 1011	51	1000 1101	93

نلاحظ في المثالين السابقين عدم التطابق بين الصورة العشرية والصورة الثانية ولكن الظريف في الأمر أننا يمكن أن نحصل على التطابق المطلوب بمجرد إضافة الرقم 6 (0110) إلى النتيجة كما يلى :

DCB	عشري	BCD	عشري
1001 0010	29	0101 0101	55
0010 0010 +	22 +	0011 1000 +	38 +
<u>0100 1011</u>	<u>51</u>	<u>1000 1101</u>	<u>93</u>
0110 +		0110 +	
<u>0101 0001</u>		<u>1001 0011</u>	<u>(صح)</u>

من ذلك نخرج بنتيجة مهمة وهي أنه في حالة جمع رقمين كل منهما 4 بتات فإنه إذا كانت النتيجة أقل من أو تساوى 9 فإن هذه النتيجة تكون متطابقة مع النظام العشري ولا تحتاج لعملية ضبط adjust. أما إذا كانت النتيجة أكبر من 9 أو كان هناك حمل من الخانة الثالثة (الأخيرة) فإنه للحصول على النتيجة في الصورة العشرية فلابد من عملية ضبط بإضافة الرقم 6 (0110) للنتيجة . لزيادة التأكيد على ذلك انظر للأمثلة التالية :

1001	9	0100	4	0011	3
1001 +	9 +	1001 +	9 +	0101 +	5 +
10010	18	1101	13	1000	8
0110+		0110 +			
11000		1 0011			

أما في حالة جمع رقمين كل منهما 8 بتات فإن نفس النتيجة السابقة تطبق على كل نصف من النتيجة على حده . أى أنه إذا كان النصف الأول من النتيجة أكبر من 9 أو كان هناك حمل من الخانة الثالثة إلى الخانة الرابعة (أى أن علم الحمل الثنائى HCF يساوى واحدا) فإنه يجب إضافة الرقم 6 (0110) إلى النتيجة كعملية ضبط . وأما إذا كان النصف الثاني من النتيجة أكبر من 9 أو حصل حمل من الخانة السابعة (أى علم الحمل يساوى واحد) فإنه يجب إضافة الرقم H 60H (0110 0000) إلى النتيجة كعملية ضبط للحصول على النتيجة في الصورة العشرية . الأمثلة التالية توضح ذلك :

$$\begin{array}{r}
 1001\ 0101 & 59 \\
 0011\ 1001 + & 39 + \\
 \hline
 1001\ 0010 & 98 \\
 \text{لذلك لزم إضافة } 0110 & \text{إلى النتيجة.} \\
 0110 + & \\
 \hline
 1001\ 1000 &
 \end{array}$$

$$\begin{array}{r}
 1000\ 1000 & 88 \\
 0100\ 1001 + & 49 + \\
 \hline
 1101\ 0001 & 137 \\
 \text{لذلك لزم إضافة } 0110 & \text{إلى النتيجة.} \\
 0110 + & \\
 \hline
 1101\ 0111 & \\
 \text{النصف الأخير من النتيجة أكبر من 9} & \\
 \text{لذلك لزم إضافة } 0110\ 0000 & \text{إلى النتيجة.} \\
 0110\ 0000 + & \\
 \hline
 1\ 0011\ 0111 &
 \end{array}$$

إن عملية إضافة الرقم 6 أو $60H$ إلى النتيجة تسمى عملية الضبط العشري وهذه العملية يقوم بها الكثير من المعالجات بناء على الأمر DAA والذي يعني Dicimal Adjust accumulator after Addition عملية الجمع .

يمكن إجراء نفس عملية الضبط السابقة بعد عملية الطرح ، الإختلاف هو أننا نطرح الرقم 6 من النتيجة إذا كان النصف الأول منها أكبر من 9 أو كان علم الحمل البيني HC يساوى واحدا ، ونطرح الرقم $60H$ من النتيجة إذا كان النصف الثاني منها أكبر من 9 أو كان علم الحمل CF يساوى واحدا ، الأمثلة التالية توضح ذلك :

$$\begin{array}{r}
 1000\ 0110 & 86 \\
 0101\ 0111 - & 57 - \\
 \hline
 0010\ 1111 & 29 \\
 \text{النصف الأول من النتيجة أكبر من 9 لذلك لزم} & \\
 \text{طرح الرقم 6 منها .} & \\
 0110 - & \\
 \hline
 0010\ 1001 &
 \end{array}$$

$$\begin{array}{r}
 & 1001\ 0100 & 49 \\
 & 0111\ 0010 & - 72 \\
 \hline
 & 1101\ 0111 & - 77 \\
 & 0110\ 0000 & \\
 \hline
 & 0111\ 0111 &
 \end{array}$$

النصف الثاني من النتيجة أكبر من 9 لذلك لزم
طرح الرقم 60H منها .

لنسى كون النتيجة يجب أن تكون سالبة أو موجبة الآن ونعلم أن عملية طرح الرقم 6 أو 60H من النتيجة يقوم بها الكثير من المعالجات بناء على الأمر DAS أي Decimal Adjust accumulator after Subtraction والتي تعنى ضبط المركم عشريا بعد عملية الطرح .

تمثيل الأرقام السالبة في النظام الثنائي

في النظام العشري نستطيع تمييز الأرقام السالبة بوضع إشارة (-) قبل الرقم كما نستطيع تمييز الأرقام الموجبة بوضع الإشارة (+) أمامه . أما في النظام الثنائي حيث الواحد والصفر هما الشكلان الوحيدان المتاحان للاستخدام فلابد وأن يكون هناك وسيلة لتمثيل الأرقام السالبة والموجبة . ولقد تم التعارف على أن تكون آخر بت في الرقم تمثل إشارة ذلك الرقم وتم التعارف أيضا على أنه إذا كانت آخر بت تساوى صفرًا فإن ذلك الرقم يكون موجبا وإذا كانت آخر بت تساوى واحدا فإن ذلك الرقم يكون سالبا . وإليك بعض الأرقام السالبة والموجبة على حسب التعارف السابق :

خانة الإشارة		
+7	0	0000111
+46	0	0101110
+64	0	1000000
-12	1	1110100
-46	1	1010010

المتمم الثنائي

لتسهيل العمليات الحسابية ، وبالذات عمليات الطرح ، في ظل هذا التعارف السابق فقد تم استخدام نظام المتمم الثنائي لتمثيل الأرقام السالبة . المتمم الثنائي لأى رقم ثانى يمكن الحصول عليه بعكس كل بت فى الرقم (جعل الواحد صفرًا

والصفر واحدا) ثم إضافة واحد لنتيجة هذا العكس . هذا المتمم الثنائي للرقم يمثل الرقم سالبا . الأمثلة التالية توضح ذلك :

- الرقم $7+7$ يمثل ثنائيا بالرقم 00000111 حيث نلاحظ أن آخر بت تساوى صفراء دلالة على أن هذا الرقم موجب . اذن كيف نمثل الرقم $7-7$ ؟
- للحصول على الرقم $7-7$ الثنائي نتبع الخطوات التالية :

الرقم $7+7$ هو 00000111
 اعكس الرقم 11111000
 أضاف واحدا 11111001

هذه النتيجة الأخيرة (11111001) تمثل الرقم $7-7$ في النظام الثنائي .
 لاحظ أنه في ظل تخصيص الخانة الأخيرة للإشارة أصبحت قيمة الرقم ممثلة فقط بسبعين (بتاتي) ، أى أن قيمة الرقم قد نقصت حيث كانت قيمته تتراوح بين الصفر و 255 قبل اعتبار الخانة الأخيرة كخانة إشارة ، أما في ظل اعتبار الخانة الأخيرة كخانة إشارة فقد أصبحت قيمة الرقم تتراوح بين الصفر و $7+127$ للأرقام الموجبة (البت الثامنة صفر) ، وتتراوح قيمة الرقم بين -1 و -128 للأرقام السالبة (بت الإشارة واحد) . وعلى ذلك يمكن كتابة هذه الأرقام كما يلى:

01111111	+127
01111110	+126
01111101	+125
.....	
00000001	+1
00000000	0
11111111	-1
11111110	-2
.....	
10000001	-127
10000000	-128

افترض أن أمامك رقمًا وتريد معرفة قيمة هذا الرقم وهل هو سالب أم موجب ؟
 في هذه الحالة عليك أولا النظر إلى خانة الإشارة فإذا كانت صفرًا فإن هذا الرقم موجب وتحدد قيمته بباقي الخانات السبعة . أما إذا كانت خانة الإشارة تحتوى واحدا فإن ذلك يعني أن هذا الرقم سالبا وتحدد قيمته بعد حساب المتمم الثنائي للرقم . كمثال على ذلك افترض أن لديك الرقم 11111001 ، هذا الرقم سالب لأن آخر خانة تساوى واحدا ، وللحصول على قيمة هذا الرقم نحسب المتمم الثنائي له كالتالى :

الرقم 1111001
 اعكس 00000110
 أصف واحدا 00000111
 إذن هذا الرقم هو -7

إليك الآن بعض الأمثلة على عمليات الجمع والطرح للأرقام ذات الإشارة :

$$\begin{array}{r}
 00001101 +13 \\
 00001001 +9 \\
 \hline
 00010110 +22
 \end{array}$$

آخر بت في النتيجة تساوى صفرًا لذلك فالنتيجة موجبة وتساوي 22.

لاحظ أننا نتعامل في النظام الثنائي وليس النظام السبع عشرى .

$$\begin{array}{r}
 00001101 +13 \\
 \text{المتمم الثنائى للرقم 9} \quad 11110111 -9 \\
 \hline
 100000100 +4
 \end{array}$$

تم إهمال الحمل الناتج ، وطالما أن آخر بت في النتيجة تساوى صفرًا فالنتيجة موجبة وتساوي 4 .

$$\begin{array}{r}
 00001001 +9 \\
 \text{المتمم الثنائى للرقم 13} \quad 11110011 -13 \\
 \hline
 11111100 -4
 \end{array}$$

آخر بت تساوى واحد فالنتيجة سالبة ، خذ المتمم الثنائى للنتيجة وهو 00000100 تحصل على النتيجة النهائية ، لذلك فالنتيجة النهائية تساوى -4 .

$$\begin{array}{r}
 11110011 -13 \\
 11110111 -9 \\
 \hline
 111101010 -22 \\
 00010110 -22
 \end{array}$$

إهمال الحمل ، خذ المتمم الثنائى للنتيجة وهو لذلك فالنتيجة تساوى -22 .

في الأمثلة السابقة كنا نتعامل في النظام العشري أو النظام الثنائي ، ماذا سيكون الموقف عند التعامل مع النظام السبع عشرى الشائع الاستخدام في الكثير من أنظمة المعالج . إليك بعض الأمثلة التي توضح ذلك :

$$\begin{array}{r}
 00010011 \\
 11110111 \\
 \hline
 100001010
 \end{array}
 \quad
 \begin{array}{r}
 +13H \\
 -9H \\
 \hline
 +0AH
 \end{array}$$

إهمال الحمل والنتيجة هي $+0A$ لأن آخر بت في النتيجة تساوى صفراء .

$$\begin{array}{r}
 00001001 \\
 11101101 \\
 \hline
 11110110
 \end{array}
 \quad
 \begin{array}{r}
 +9H \\
 -13H \\
 \hline
 -0AH
 \end{array}$$

المتم الثنائى للرقم $13H$ (00010011) آخر بت تساوى 1 فالنتيجة سالبة ، خذ المتم الثنائى لذلك فالنتيجة هي $-0A$.

من ذلك نرى أن عملية تمثيل الأرقام السالبة بالمتم الثنائى للرقم هي نفسها فى أى من النظامين العشري أو السعشرى ، كل ما هناك هو ملاحظة الفرق فى كيفية وضع الرقم فى صورته الثنائية . إن الأمثلة السابقة توضح لنا أن عملية الطرح ما هي إلا عملية جمع للمطروح منه مع المتم الثنائى للمطروح وذلك ما تقوم به عادة دوائر المعالج التى تقوم بتنفيذ عملية الطرح كما رأينا فى الفصل السابع .

الضرب و القسمة ثنائية

يمكن إجراء الضرب الثنائى بأكثر من طريقة أولها هي طريقة الضرب فى النظام العشري والتى نعرفها جميعاً منذ الصغر حيث يتم ضرب الخانة الأولى من المضروب فى المضروب فيه وتدون النتيجة ، ثم يتم ضرب الخانة الثانية من المضروب فى المضروب فيه وتدون النتيجة تحت النتيجة السابقة ولكن تزاح لليسار بمقدار خانة ، وهكذا يتم ضرب جميع خانات المضروب فى المضروب فيه وفي كل مرة تدون النتيجة تحت النتيجة السابقة بعد إزاحتها بمقدار خانة واحدة ، وفي النهاية يتم جمع جميع النتائج السابقة كما يلى :

$$\begin{array}{r}
 1011 \text{ المضروب فيه} \\
 1001 \text{ المضروب} \\
 \hline
 1011 \\
 0000 \\
 0000 \\
 1011 \\
 \hline
 1100011
 \end{array}
 \quad
 \begin{array}{r}
 11 \\
 x 9 \\
 \hline
 99
 \end{array}$$

من الطرق الأخرى للضرب استخدام طريقة الجمع المتكرر حيث فى المثال السابق مثلاً نقوم بجمع العدد 11 مع نفسه 9 مرات . من عيوب هذه الطريقة أنها تكون بطيئة بالذات فى حالة ضرب الأرقام الكبيرة ، فمثلاً فى حالة ضرب

الرقمين 165 في 99 سنقوم بجمع الرقم 165 مع نفسه 99 مرة وذلك بالطبع يتطلب الكثير من الوقت .

في المثال السابق (11×9) نرى أن نتيجة ضرب أي خانة من المضرب في المضروب فيه تكون إما صفرًا أو المضروب فيه نفسه حيث أن هذه الخانة تكون إما صفرًا أو واحدًا . يمكن الاستفادة من ذلك في استنتاج طريقة أخرى للضرب أسرع وأنسب من الطريقيتين السابقتين كما سنرى في المثال التالي : $(13 \times 13 = 169)$

طالما أن البت الأولى من المضرب 1 كتبنا المضروب فيه كما هو .

ثم أزحنا النتيجة السابقة لليمين بمقدار خانة واحدة .
طالما أن البت الثانية من المضرب 0 لم نفعل شيء فقط أزحنا النتيجة السابقة لليمين بمقدار خانة .

طالما أن الخانة الثالثة من المضرب 1 جمعنا المضرب فيه على النتيجة السابقة فنحصل على النتيجة التالية :

ثم أزحنا النتيجة السابقة ناحية اليمين بمقدار خانة .

طالما أن البت الرابعة في المضرب 1 نجمع المضروب فيه على النتيجة السابقة فنحصل على النتيجة النهائية التالية :

$$\begin{array}{r}
 1101 \\
 1101 \times \\
 \hline
 1101 \\
 01101 \\
 001101 \\
 \hline
 001101 \\
 1101 + \\
 \hline
 1000001 \\
 01000001 \\
 \hline
 01000001 \\
 1101 + \\
 \hline
 10101001 \\
 169
 \end{array}$$

أى أن هذه الطريقة تتلخص في أننا نضرب الخانة الأولى من المضرب في المضروب فيه ثم نزير النتيجة ناحية اليمين بمقدار خانة ، وإذا كانت الخانة الثانية صفرًا فلا نعمل شيئاً سوى الإزاحة لليمين بمقدار خانة ، وإذا كانت واحدًا نجمع المضروب فيه مع النتيجة السابقة ثم نزير نتيجة الجمع السابقة خانة وننظر في الخانة الثالثة من المضرب ، وهكذا . نلاحظ أن هذه الطريقة أسرع بكثير من الطرق السابقة كما نلاحظ أن النتيجة النهائية للضرب تشغّل عدداً من البتات يساوي مجموع عدد بتات المضرب والمضروب فيه .

لإجراء عملية القسمة الثنائية نتبع نفس الخطوات التي اتبعناها في إجراء عمليات الجمع الثنائي ، سوى أننا نستخدم الطرح المتكرر بدلاً من الجمع المتكرر كما في حالة الجمع ، أو نستخدم الطرح ثم الإزاحة ناحية اليسار بدلاً من الجمع ثم الإزاحة ناحية اليمين كما في حالة الضرب .

مراجع الكتاب

1. The MFA microcomputer training kit manuals , 1986.
2. Heathkit manual for the microprocessor trainer model ET-3400, 1981.
3. Ramesh S. Gaonkar, "Microprocessor architecture, programming and applications with the 8085/8080A." Wiley Eastern Limited, 1986.
4. Douglas V. Hall "Microprocessors and digital systems" McGraw Hill Book company,1983.
5. Kathe Spracklen "Z-80 and 8080 assembly language programming" Hayden Book Company, Inc, 1979.
6. Charles K. Adams "Master handbook of microprocessor chips" TAB Books Inc. 1981.
7. David F. Stout "Microprocessor applications handbook" McGraw.Hill Book Company, 1985.
8. James W. Coffern "Z-80 Applications" 1988.
9. Lance A. Lventhal "6800 assembly language programming" Osborne & Associates Inc. 1978.
10. Joseph J. Carr "Microprocessor Interfacing" TAB Books Inc. 1982.
11. Frank P. Tedeschi & Robert Colen "101 projects for the Z-80" TAB Books Inc. 1983.
12. Hermann Schmid "Electronic analog/digital conversions" Van Nostrand Reinhold Company, 1970.
13. Micro-tech Publication "Microprocessor Data Hand Book" 1991
14. Michael Thorne "Programming the 8086/8088 for the IBM PC and Compatables" The Benjamin P. Company 1986
15. الطبعة الأولى 1991 "البرمجة بلغة التجميع" د. عوض منصور .
16. Barry B. Brey "The Intel Microprocessors 8086, 80186, 80286, 80386, 80486" Maxwell International 1991 .
17. Hans-Peter Messmer "The Indispensable PC Hardware Book" Addison- Wesley 1997 .
18. Daniel Tabak "Advanced Microprocessors" McGraw Hill Inc. 1995

المعالجات الدقيقة

Microprocessors

تأليف / د. محمد إبراهيم العدوى

— هذا الكتاب —

ما هو الميكروبيوسور (المعالج الدقيق) ؟ أين يقع المعالج في الميكروكمبيوتر ؟ كيف يمكن برمجته ؟ كيف يمكن توصيله مع الذاكرة ؟ كيف يمكن إدخال بيانات إليه، وكيف يمكن إخراج بيانات منه ؟ كيف يمكن استخدام المعالج كعنصر أساسى في دوائر التحكم مثل التحكم في درجات الحرارة والضغط وإشارات المرور وغيرها ؟ ما هو الميكروكمبيوتر ذو الكارت الواحد ؟ ما هي البرامج الفرعية ؟ كيف يمكن مقاطعة الميكروبيوسور ؟ كل هذه الأسئلة وأكثر يجيب عنها هذا الكتاب في أسلوب سهل وأفكار متدرجة خالية من التعقيد ومن خلال أمثلة ومقارنات كثيرة مطبقة على أشهر المعالجات ذات 8 بت وهي المعالج intel 8085 والمعالج Z80 وكذلك أشهر المعالجات ذات 16 بت وهو المعالج intel 8086 ولن يقف الكتاب عند هذا الحد بل قدم المعالجات الأخرى مثل المعالجات 80186 و 80286 و 80386 و 80486 وحتى الأجيال التالية من عائلة تشيم ..

Bibliotheca Alexandrina



0354220

IHCI

ATIONAL HOUSE FOR CULTURAL INVESTMENTS

CAIRO - EGYPT

ISBN : 977-282-076-5