

Nama : Maya Nurkhayati
NIM : L200210220
Kelas : E

TUGAS MODUL 1

1. Apa yang dimaksud kode 'ASCII', buatlah tabel kode ASCII lengkap cukup kode ASCII yang standar tidak perlu extended, tuliskan kode ASCII dalam format angka, decimal, binary dan hexadecimal serta karakter dan symbol yang dikodekan.

ASCII (American Standard Code for Information Interchange) merupakan Kode Standar Amerika untuk Pertukaran Informasi atau sebuah standar internasional dalam pengkodean huruf dan simbol seperti Unicode dan Hex tetapi ASCII lebih bersifat universal.

Nilai ANSI ASCII (Desimal)	Nilai Unicode (Heksa Desimal)	Karakter	Keterangan
0	0000	NUL	Null (tidak terlihat)
1	0001	SOH	Start of heading (tidak terlihat)
2	0002	STX	Start of text (tidak terlihat)
3	0003	ETX	End of text (tidak terlihat)
4	0004	EOT	End of transmission (tidak terlihat)
5	0005	ENQ	Enquiry (tidak terlihat)
6	0006	ACK	Acknowledge (tidak terlihat)
7	0007	BEL	Bell (tidak terlihat)
8	0008	BS	Backspace
9	0009	HT	Horizontal tabulation
10	000A	LF	Pergantian baris (Line feed)
11	000B	VT	Tabulasi vertical
12	000C	FF	Pergantian baris (Form feed)
13	000D	CR	Pergantian baris (carriage return)

14	000E	SO	Shift out (tidak terlihat)
15	000F	SI	Shift in (tidak terlihat)
16	0010	DLE	Data link escape (tidak terlihat)
17	0011	DC1	Device control 1 (tidak terlihat)
18	0012	DC2	Device control 2 (tidak terlihat)
19	0013	DC3	Device control 3 (tidak terlihat)
20	0014	DC4	Device control 4 (tidak terlihat)
21	0015	NAK	Negative acknowledge (tidak terlihat)
22	0016	SYN	Synchronous idle (tidak terlihat)
23	0017	ETB	End of transmission block (tidak terlihat)
24	0018	CAN	Cancel (tidak terlihat)
25	0019	EM	End of medium (tidak terlihat)
26	001A	SUB	Substitute (tidak terlihat)
27	001B	ESC	Escape (tidak terlihat)
28	001C	FS	File separator
29	001D	GS	Group separator
30	001E	RS	Record separator
31	001F	US	Unit separator
32	0020	spasi	Spasi
33	0021	!	Tanda seru (exclamation)
34	0022	“	Tanda kutip dua
35	0023	#	Tanda pagar (kres)

36	0024	\$	Tanda mata uang dolar
37	0025	%	Tanda persen
38	0026	&	Karakter ampersand (&)
39	0027	'	Karakter Apostrof
40	0028	(Tanda kurung buka
41	0029)	Tanda kurung tutup
42	002A	*	Karakter asterisk (bintang)
43	002B	+	Tanda tambah (plus)
44	002C	,	Karakter koma
45	002D	–	Karakter hyphen (strip)
46	002E	.	Tanda titik
47	002F	/	Garis miring (slash)
48	0030	0	Angka nol
49	0031	1	Angka satu
50	0032	2	Angka dua
51	0033	3	Angka tiga
52	0034	4	Angka empat
53	0035	5	Angka lima
54	0036	6	Angka enam
55	0037	7	Angka tujuh
56	0038	8	Angka delapan
57	0039	9	Angka Sembilan
58	003A	:	Tanda titik dua

59	003B	;	Tanda titik koma
60	003C	<	Tanda lebih kecil
61	003D	=	Tanda sama dengan
62	003E	>	Tanda lebih besar
63	003F	?	Tanda tanya
64	0040	@	A keong (@)
65	0041	A	Huruf latin A kapital
66	0042	B	Huruf latin B kapital
67	0043	C	Huruf latin C kapital
68	0044	D	Huruf latin D kapital
69	0045	E	Huruf latin E kapital
70	0046	F	Huruf latin F kapital
71	0047	G	Huruf latin G kapital
72	0048	H	Huruf latin H kapital
73	0049	I	Huruf latin I kapital
74	004A	J	Huruf latin J kapital
75	004B	K	Huruf latin K kapital
76	004C	L	Huruf latin L kapital
77	004D	M	Huruf latin M kapital
78	004E	N	Huruf latin N kapital
79	004F	O	Huruf latin O kapital
80	0050	P	Huruf latin P kapital
81	0051	Q	Huruf latin Q kapital

82	0052	R	Huruf latin R capital
83	0053	S	Huruf latin S capital
84	0054	T	Huruf latin T capital
85	0055	U	Huruf latin U capital
86	0056	V	Huruf latin V capital
87	0057	W	Huruf latin W capital
88	0058	X	Huruf latin X capital
89	0059	Y	Huruf latin Y capital
90	005A	Z	Huruf latin Z capital
91	005B	[Kurung siku kiri
92	005C	/	Garis miring terbalik (<i>backslash</i>)
93	005D]	Kurung sikur kanan
94	005E	^	Tanda pangkat
95	005F	—	Garis bawah (underscore)
96	0060	`	Tanda petik satu
97	0061	a	Huruf latin a kecil
98	0062	b	Huruf latin b kecil
99	0063	c	Huruf latin c kecil
100	0064	d	Huruf latin d kecil
101	0065	e	Huruf latin e kecil
102	0066	f	Huruf latin f kecil
103	0067	g	Huruf latin g kecil
104	0068	h	Huruf latin h kecil

105	0069	i	Huruf latin i kecil
106	006A	j	Huruf latin j kecil
107	006B	k	Huruf latin k kecil
108	006C	l	Huruf latin l kecil
109	006D	m	Huruf latin m kecil
110	006E	n	Huruf latin n kecil
111	006F	o	Huruf latin o kecil
112	0070	p	Huruf latin p kecil
113	0071	q	Huruf latin q kecil
114	0072	r	Huruf latin r kecil
115	0073	s	Huruf latin s kecil
116	0074	t	Huruf latin t kecil
117	0075	u	Huruf latin u kecil
118	0076	v	Huruf latin v kecil
119	0077	w	Huruf latin w kecil
120	0078	x	Huruf latin x kecil
121	0079	y	Huruf latin y kecil
122	007A	z	Huruf latin z kecil
123	007B	{	Kurung kurawal buka
124	007C		Garis vertikal (pipa)
125	007D	}	Kurung kurawal tutup
126	007E	~	Karakter gelombang (tilde)
127	007F	DEL	Delete

128	0080	€	Euro sign
129	0081		
130	0082	,	Single low-9 quotation mark
131	0083	<i>f</i>	Latin small letter f with hook
132	0084	„	Double low-9 quotation mark
133	0085	...	Horizontal ellipsis
134	0086	†	Dagger
135	0087	‡	Double dagger
136	0088	^	Modifier letter circumflex accent
137	0089	‰	Per mille sign
138	008A	Š	Latin capital letter S with caron
139	008B	‹	Single left-pointing angle quotation
140	008C	Œ	Latin capital ligature OE
141	008D		
142	008E	Ž	Latin captial letter Z with caron
143	008F		
144	0090		
145	0091	‘	Left single quotation mark
146	0092	’	Right single quotation mark
147	0093	“	Left double quotation mark
148	0094	”	Right double quotation mark
149	0095	•	Bullet
150	0096	—	En dash

151	0097	—	Em dash
152	0098	~	Small tilde
153	0099	™	Trade mark sign
154	009A	š	Latin small letter S with caron
155	009B	›	Single right-pointing angle quotation mark
156	009C	œ	Latin small ligature oe
157	009D		
158	009E	ž	Latin small letter z with caron
159	009F	ÿ	Latin capital letter Y with diaeresis
160	00A0		Spasi yang bukan pemisah kata
161	00A1	¡	Tanda seru terbalik
162	00A2	¢	Tanda sen (Cent)
163	00A3	£	Tanda Poundsterling
164	00A4	¤	Tanda mata uang (Currency)
165	00A5	¥	Tanda Yen
166	00A6	¦	Garis tegak putus-putus
167	00A7	§	Section sign
168	00A8	¨	Spacing diaeresis – umlaut
169	00A9	©	Tanda hak cipta (Copyright)
170	00AA	ª	Feminine ordinal indicator
171	00AB	«	Left double angle quotes
172	00AC	¬	Not sign
173	00AD		Tanda strip (hyphen)

174	00AE	®	Tanda merk terdaftar
175	00AF	ˉ	Spacing Macron (Macron)
176	00B0	°	Tanda derajat
177	00B1	±	Tanda kurang lebih (plus-minus)
178	00B2	²	Tanda kuadrat (pangkat dua)
179	00B3	³	Tanda kubik (pangkat tiga)
180	00B4	´	Acute accent
181	00B5	μ	Micro sign
182	00B6	¶	Pilcrow sign
183	00B7	·	Middle dot
184	00B8	¸	Spacing cedilla
185	00B9	¹	Superscript one
186	00BA	º	Masculine ordinal indicator
187	00BB	»	Right double angle quotes
188	00BC	¼	Fraction one quarter
189	00BD	½	Fraction one half
190	00BE	¾	Fraction three quarters
191	00BF	¿	Inverted question mark
192	00C0	À	Latin capital letter A with grave
193	00C1	Á	Latin capital letter A with acute
194	00C2	Â	Latin capital letter A with circumflex
195	00C3	Ã	Latin capital letter A with tilde
196	00C4	Ä	Latin capital letter A with diaeresis

197	00C5	À	Latin capital letter A with ring above
198	00C6	Æ	Latin capital letter AE
199	00C7	Ç	Latin capital letter C with cedilla
200	00C8	È	Latin capital letter E with grave
201	00C9	É	Latin capital letter E with acute
202	00CA	Ê	Latin capital letter E with circumflex
203	00CB	Ë	Latin capital letter E with diaeresis
204	00CC	Ì	Latin capital letter I with grave
205	00CD	Í	Latin capital letter I with acute
206	00CE	Î	Latin capital letter I with circumflex
207	00CF	Ï	Latin capital letter I with diaeresis
208	00D0	Ð	Latin capital letter ETH
209	00D1	Ñ	Latin capital letter N with tilde
210	00D2	Ò	Latin capital letter O with grave
211	00D3	Ó	Latin capital letter O with acute
212	00D4	Ô	Latin capital letter O with circumflex
213	00D5	Õ	Latin capital letter O with tilde
214	00D6	Ö	Latin capital letter O with diaeresis
215	00D7	×	Multiplication sign
216	00D8	Ø	Latin capital letter O with slash
217	00D9	Ù	Latin capital letter U with grave
218	00DA	Ú	Latin capital letter U with acute
219	00DB	Û	Latin capital letter U with circumflex

220	00DC	Ü	Latin capital letter U with diaeresis
221	00DD	Ý	Latin capital letter Y with acute
222	00DE	Þ	Latin capital letter THORN
223	00DF	ß	Latin small letter sharp s – ess-zed
224	00E0	à	Latin small letter a with grave
225	00E1	á	Latin small letter a with acute
226	00E2	â	Latin small letter a with circumflex
227	00E3	ã	Latin small letter a with tilde
228	00E4	ä	Latin small letter a with diaeresis
229	00E5	å	Latin small letter a with ring above
230	00E6	æ	Latin small letter ae
231	00E7	ç	Latin small letter c with cedilla
232	00E8	è	Latin small letter e with grave
233	00E9	é	Latin small letter e with acute
234	00EA	ê	Latin small letter e with circumflex
235	00EB	ë	Latin small letter e with diaeresis
236	00EC	ì	Latin small letter i with grave
237	00ED	í	Latin small letter i with acute
238	00EE	î	Latin small letter i with circumflex
239	00EF	ï	Latin small letter i with diaeresis
240	00F0	ð	Latin small letter eth
241	00F0	ñ	Latin small letter n with tilde
242	00F0	ò	Latin small letter o with grave

243	00F0	ó	Latin small letter o with acute
244	00F0	ô	Latin small letter o with circumflex
245	00F0	õ	Latin small letter o with tilde
246	00F0	ö	Latin small letter o with diaeresis
247	00F0	÷	Division sign
248	00F0	ø	Latin small letter o with slash
249	00F0	ù	Latin small letter u with grave
250	00F0	ú	Latin small letter u with acute
251	00F0	û	Latin small letter u with circumflex
252	00F0	ü	Latin small letter u with diaeresis
253	00F0	ý	Latin small letter y with acute
254	00F0	þ	Latin small letter thorn
255	00F0	ÿ	Latin small letter y with diaeresis

2. Carilah daftar perintah bahasa assembly untuk mesin intel keluarga x86 lengkap(dari buku referensi atau internet). Daftar perintah ini dapat digunakan sebagai pedoman untuk memahami program 'boot.asm' dan 'kernel.am'.

1. ACALL (Absolute Call)

ACALL berfungsi untuk memanggil sub rutin program

2. ADD (Add Immediate Data)

ADD berfungsi untuk menambah 8 bit data langsung ke dalam isi akumulator dan menyimpan hasilnya pada akumulator.

3. ADDC (Add Carry Plus Immediate Data to Accumulator)

ADDC berfungsi untuk menambahkan isi carry flag (0 atau 1) ke dalam isi akumulator. Data langsung 8 bit ditambahkan ke akumulator.

4. AJMP (Absolute Jump)

AJMP adalah perintah jump mutlak. Jump dalam 2 KB dimulai dari alamat yang mengikuti perintah AJMP. AJMP berfungsi untuk mentransfer kendali program ke lokasi dimana alamat dikalkulasi dengan cara yang sama dengan perintah ACALL. Konter program ditambahkan dua kali dimana perintah AJMP adalah perintah 2-byte. Konter program di-load dengan a10 – a0 11 bits, untuk membentuk alamat tujuan 16-bit.

5. ANL (logical AND memori ke akumulator)

ANL berfungsi untuk mengAND-kan isi alamat data dengan isi akumulator.

6. CJNE (Compare Indirect Address to Immediate Data)

CJNE berfungsi untuk membandingkan data langsung dengan lokasi memori yang dialamati oleh register R atau Akumulator A. apabila tidak sama maka instruksi akan menuju ke alamat kode.

Format : CJNE R,#data,Alamat kode.

7. CLR (Clear Accumulator)

CLR berfungsi untuk mereset data akumulator menjadi 00H.

Format : CLR A

8. CPL (Complement Accumulator)

CPL berfungsi untuk mengkomplemen isi akumulator.

9. DA (Decimal Adjust Accumulator)

DA berfungsi untuk mengatur isi akumulator ke padanan BCD, setelah penambahan dua angka BCD.

10. DEC (Decrement Indirect Address)

DEC berfungsi untuk mengurangi isi lokasi memori yang ditujukan oleh register R dengan 1, dan hasilnya disimpan pada lokasi tersebut.

11. DIV (Divide Accumulator by B)

DIV berfungsi untuk membagi isi akumulator dengan isi register B. Akumulator berisi hasil bagi, register B berisi sisa pembagian.

12. DJNZ (Decrement Register And Jump If Not Zero)

DJNZ berfungsi untuk mengurangi nilai register dengan 1 dan jika hasilnya sudah 0 maka instruksi selanjutnya akan dieksekusi. Jika belum 0 akan menuju ke alamat kode.

13. INC (Increment Indirect Address)

INC berfungsi untuk menambahkan isi memori dengan 1 dan menyimpannya pada alamat tersebut.

14. JB (Jump if Bit is Set)

JB berfungsi untuk membaca data per satu bit, jika data tersebut adalah 1 maka akan menuju ke alamat kode dan jika 0 tidak akan menuju ke alamat kode.

15. JBC (Jump if Bit Set and Clear Bit)

Bit JBC, berfungsi sebagai perintah rel menguji yang terspesifikasikan secara bit. Jika bit di-set, maka Jump dilakukan ke alamat relatif dan yang terspesifikasi secara bit di dalam perintah dibersihkan. Segmen program berikut menguji bit yang kurang signifikan (LSB: Least Significant Byte), dan jika diketemukan bahwa ia telah di-set, program melompat ke READ lokasi. JBC juga berfungsi membersihkan LSB dari akumulator.

16. JC (Jump if Carry is Set)

Instruksi JC berfungsi untuk menguji isi carry flag. Jika berisi 1, eksekusi menuju ke alamat kode, jika berisi 0, instruksi selanjutnya yang akan dieksekusi.

17. JMP (Jump to sum of Accumulator and Data Pointer)

Instruksi JMP berfungsi untuk memerintahkan loncat kesuatu alamat kode tertentu.
Format : JMP alamat kode.

18. JNB (Jump if Bit is Not Set)

Instruksi JNB berfungsi untuk membaca data per satu bit, jika data tersebut adalah 0 maka akan menuju ke alamat kode dan jika 1 tidak akan menuju ke alamat kode.
Format : JNB alamat bit, alamat kode.

19. JNC (Jump if Carry Not Set)

JNC berfungsi untuk menguji bit Carry, dan jika tidak di-set, maka sebuah lompatan akan dilakukan ke alamat relatif yang telah ditentukan.

20. JNZ (Jump if Accumulator Not Zero)

JNZ adalah mnemonik untuk instruksi jump if not zero (lompat jika tidak nol). Dalam hal ini suatu lompatan akan terjadi bilamana bendera nol dalam keadaan "clear", dan tidak akan terjadi lompatan bilamana bendera nol tersebut dalam keadaan set. Andaikan bahwa JNZ 7800H disimpan pada lokasi 2100H. Jika Z=0, instruksi berikutnya akan berasal dari lokasi 7800H: dan bilamana Z=1, program akan turun ke instruksi urutan berikutnya pada lokasi 2101H.

21. JZ (Jump if Accumulator is Zero)

JZ berfungsi untuk menguji konten-konten akumulator. Jika bukan nol, maka lompatan dilakukan ke alamat relatif yang ditentukan dalam perintah.

22. LCALL (Long Call)

LCALL berfungsi untuk memungkinkan panggilan ke subrutin yang berlokasi dimanapun dalam memori program 64K. Operasi LCALL berjalan seperti berikut:

- Menambahkan ke dalam konter program sebanyak 3, karena perintahnya adalah perintah 3-byte.
- Menambahkan penunjuk stack sebanyak 1.
- Menyimpan byte yang lebih rendah dari konter program ke dalam stack.
- Menambahkan penunjuk stack.
- Menyimpan byte yang lebih tinggi dari program ke dalam stack.
- Me-load konter program dengan alamat tujuan 16-bit.

23. . Ljmp (Long Jump)

Long Jump berfungsi untuk memungkinkan lompatan tak bersyarat kemana saja dalam lingkup ruang memori program 64K. LCALL adalah perintah 3-byte. Alamat tujuan 16-bit ditentukan secara langsung dalam perintah tersebut. Alamat tujuan ini di-load ke dalam konter program oleh perintah Ljmp.

24. MOV (Move From Memory)

MOV berfungsi untuk memindahkan isi akumulator/register atau data dari nilai luar atau alamat lain.

25. MOVC (Move From Code Memory)

Instruksi MOVC berfungsi untuk mengisi accumulator dengan byte kode atau konstanta dari program memory. Alamat byte tersebut adalah hasil penjumlahan unsigned 8 bit pada accumulator dan 16 bit register basis yang dapat berupa data pointer atau program counter. Instruksi ini tidak mempengaruhi flag apapun juga.

26. MOVX (Move Accumulator to External Memory Addressed by Data Pointer)

MOVX berfungsi untuk memindahkan isi akumulator ke memori data eksternal yang alamatnya ditunjukkan oleh isi data pointer.

27. MUL (Multiply)

MUL AB berfungsi untuk mengalikan unsigned 8 bit integer pada accumulator dan register B. Byte rendah (low order) dari hasil perkalian akan disimpan dalam accumulator sedangkan byte tinggi (high order) akan disimpan dalam register B. Jika hasil perkalian lebih besar dari 255 (0FFh), overflow flag akan bernilai '1'. Jika hasil perkalian lebih kecil atau sama dengan 255, overflow flag akan bernilai '0'. Carry flag akan selalu dikosongkan.

28. NOP (No Operation)

Fungsi NOP adalah eksekusi program akan dilanjutkan ke instruksi berikutnya. Selain PC, instruksi ini tidak mempengaruhi register atau flag apapun juga.

29. ORL (Logical OR Immediate Data to Accumulator)

Instruksi ORL berfungsi sebagai instruksi Gerbang logika OR yang akan menjumlahkan Accumulator terhadap nilai yang ditentukan.

Format : ORL A,#data.

30. POP (Pop Stack to Memory)

Instruksi POP berfungsi untuk menempatkan byte yang ditunjukkan oleh stack pointer ke suatu alamat data.

31. PUSH (Push Memory onto Stack)

Instruksi PUSH berfungsi untuk menaikkan stack pointer kemudian menyimpan isinya ke suatu alamat data pada lokasi yang ditunjuk oleh stack pointer.

32. RET (Return from subroutine)

Intruksi RET berfungsi untuk kembali dari suatu subrutin program ke alamat terakhir subrutin tersebut di panggil.

33. RETI (Return From Interrupt)

RETI berfungsi untuk mengambil nilai byte tinggi dan rendah dari PC dari stack dan mengembalikan kondisi logika interrupt agar dapat menerima interrupt lain dengan prioritas yang sama dengan prioritas interrupt yang baru saja diproses. Stack pointer

akan dikurangi dengan 2. Instruksi ini tidak mempengaruhi flag apapun juga. Nilai PSW tidak akan dikembalikan secara otomatis ke kondisi sebelum interrupt. Eksekusi program akan dilanjutkan pada alamat yang diambil tersebut. Umumnya alamat tersebut adalah alamat setelah lokasi dimana terjadi interrupt. Jika interrupt dengan prioritas sama atau lebih rendah tertunda saat RETI dieksekusi, maka satu instruksi lagi akan dieksekusi sebelum interrupt yang tertunda tersebut diproses.

34. RL (Rotate Accumulator Left)

Instruksi RL berfungsi untuk memutar setiap bit dalam akumulator satu posisi ke kiri.

35. . RLC (Rotate Left through Carry)

Fungsi : Memutar (Rotate) Accumulator ke Kiri (Left) Melalui Carry Flag. Kedelapan bit accumulator dan carry flag akan diputar satu bit ke kiri secara bersama-sama. Bit 7 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 0. Instruksi ini tidak mempengaruhi flag lain.

36. RR (Rotate Right)

Fungsi : Memutar (Rotate) Accumulator ke Kanan (Right). Kedelapan bit accumulator akan diputar satu bit ke kanan. Bit 0 akan dirotasi ke posisi bit 7. Instruksi ini tidak mempengaruhi flag apapun juga.

37. RRC (Rotate Right through Carry)

Fungsi : Memutar (Rotate) Accumulator ke Kanan (Right) Melalui Carry Flag. Kedelapan bit accumulator dan carry flag akan diputar satu bit ke kanan secara bersama-sama. Bit 0 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 7. Instruksi ini tidak mempengaruhi flag lain.

38. SETB (set Carry flag)

Instruksi SETB berfungsi untuk menset carry flag.

39. SJMP (Short Jump)

Sebuah Short Jump berfungsi untuk mentransfer kendali ke alamat tujuan dalam 127 bytes yang mengikuti dan 128 yang mengawali perintah SJMP. Alamat tujuannya ditentukan sebagai sebuah alamat relative 8-bit. Ini adalah Jump tidak bersyarat. Perintah SJMP menambahkan konter program sebanyak 2 dan menambahkan alamat relatif ke dalamnya untuk mendapatkan alamat tujuan. Alamat relatif tersebut ditentukan dalam perintah sebagai 'SJMP rel'.

40. SUBB (Subtract With Borrow)

Fungsi : Pengurangan (Subtract) dengan Peminjaman (Borrow). SUBB mengurangi variabel yang tertera pada operand kedua dan carry flag sekaligus dari accumulator dan menyimpan hasilnya pada accumulator.

41. SWAP (Swap Nibbles)

Fungsi : Menukar (Swap) Upper Nibble dan Lower Nibble dalam Accumulator. SWAP A akan menukar nibble (4 bit) tinggi dan nibble rendah dalam accumulator. Operasi ini dapat dianggap sebagai rotasi 4 bit dengan RR atau RL. Instruksi ini tidak mempengaruhi flag apapun juga.

42. XCH (Exchange Bytes)

Fungsi : Menukar (Exchange) Accumulator dengan Variabel Byte. XCH akan mengisih accumulator dengan variabel yang tertera pada operand kedua dan pada saat yang

sama juga akan mengisi nilai accumulator ke dalam variabel tersebut. Addressing mode yang dapat digunakan adalah: register, direct, atau register indirect.

43. XCHD (Exchange Digits)

Fungsi : Menukar (Exchange) Digit. XCHD menukar nibble rendah dari accumulator, yang umumnya mewakili angka heksadesimal atau BCD, dengan nibble rendah dari internal data memory yang diakses secara indirect. Nibble tinggi kedua register tidak akan terpengaruh. Instruksi ini tidak mempengaruhi flag apapun juga.

44. XRL (Exclusive OR Logic)

Fungsi : Logika Exclusive OR untuk Variabel Byte XRL akan melakukan operasi bitwise logika exclusive OR antara kedua variabel yang dinyatakan. Hasilnya akan disimpan pada destination byte. Instruksi ini tidak mempengaruhi flag apapun juga. Kedua operand mampu menggunakan enam kombinasi addressing mode. Saat destination byte adalah accumulator, source byte dapat berupa register, direct, register indirect, atau immediate data. Saat destination byte berupa direct address, source byte dapat berupa accumulator atau immediate data.