

1.4 Архітектура системи команд базового мікропроцесора

Обчислювальна потужність побудованої мікропроцесорної системи визначається архітектурою головного мікропроцесора. Однією із складової архітектури мікропроцесора є, безумовно, системи команд – сукупність окремих груп, кожна із яких насичена окремою множиною інструкцій. Такими групами є команди пересилки даних, логічної та арифметичної обробки, команди передачі управління та команди керування пристроями мікропроцесора.

Арифметико - логічна обробка даних передбачає відповідні обчислення байтів, представлених беззнаковими цілими числами. Багаторозрядні числа в процесорі обробляються як елементи ланцюгів даних з урахуванням перенесень між байтами. Недоліком системи є відсутність команд обчислень знакових чисел, а також неможливість сприйняття змішаних чисел (мається на увазі чисел з дробовими частинами). Відсутність команд множення та ділення є також суттєвим недоліком. Бітова обробка замінюється складнощами, пов'язаними з програмними маскуваннями байтів даних. Наявність команди корекції після складання двійково-десяткових чисел спрощує алгоритми програмної обробки десяткових чисел в комп'ютерній системі.

Пересилка байтів даних між регістрами процесора та регістрами і пам'яттю виконується в процесорі відповідно за один та два машинних цикли. При передбаченій командою передачі слів пересилка виконується байтами за два машинних цикли. При передачі в стек спочатку записується старший байт, а потім молодший. Видача зі стеку при його читанні спроваджується читанням спочатку молодшого, а потім передачею старшого байту.

Відсутність команд переривань мікропроцесора компенсується командами повторного пуску процесора, що забезпечує перемикання процесора на окремі ареали програмної пам'яті початкових фізичних адрес. Виклики процедур в повні мірі задовольняють вимоги збереження адреси повернення в системному стеку. Велика множина команд умовного та безумовного переходів також спрощує будову складних алгоритмів.


					ДП 5.05010201 435 01 ПЗ	Арк.
						39
Змін.	Арк.	№ докум.	Підпис	Дата		

1.4.1 Група команд пересилки даних

Команди цієї групи використовують регістрову, непряму, пряму і безпосередню адресацію операнду. Перелік команд досить насичений мнемоніками.


1.1 MOV r1/M,r2/M	1:1:5/2:7	1.6 PUSH/POP rp	1:3:11/10
1.2 MVI r/M, data	2:2:7/3:10	1.7 SPHL	1:1:5
1.3 LDA/STA addr	3:4:13	1.8 XCHG/XTHL	1:1:4/18
1.4 LHLD/SHLD addr	3:5:16	1.9 IN/OUT port	2:3:10
1.5 LDAX/STAX rp 1:2:7			

Найбільш потужна є команда 1.1, із якої можна утворити три команди

MOV r1, r2;	r1:=(r2)		
MOV r, M ;	r := M(HL)		M1:FETCH - 4 M2:MEMORY_WRITE - 3
MOV M, r;	M(HL) :=(r)		

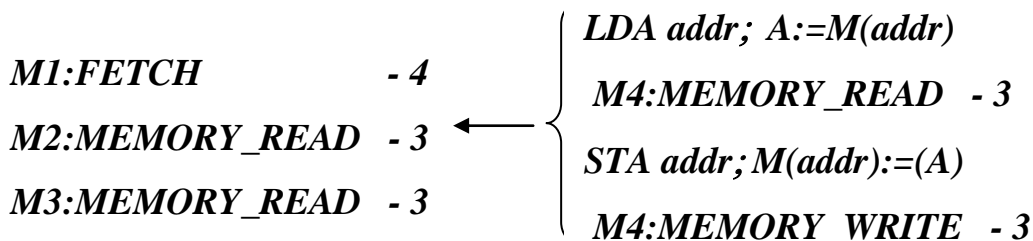
Останні дві команди двохциклові з непрямою адресацією та часом виконання в сім тактів. Перша команда регістрової адресації виконується за один цикл в п'ять тактів.

Формат двохбайтної 1.2 породжує дві інструкції завантаження безпосереднього операнду в регістр чи пам'ять, непрямо адресуєму парою регістрів **HL**.

<u>MoVe Immediate Regsster / Memory</u>			
MVI r, data;	r:=data		M1:FETCH - 4 M2:MEMORY_READ - 3 M3:MEMORY_WRITE - 3
MVI M, data ;	M(HL):= data		

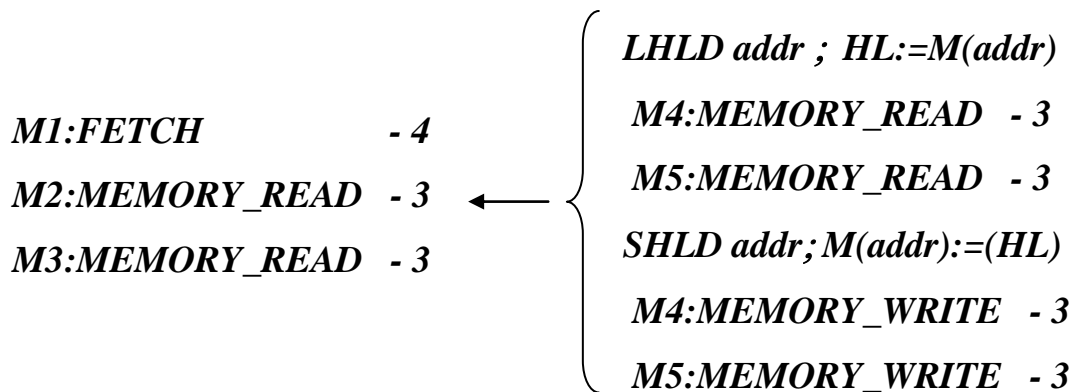
Дві команди взаємо протиположної передачі байтів між акумулятором і прямо адресованою пам'яттю даних (**Load/Store Accumulator direct ADDRESS**) дуже схожі у часу їх вибору і програмної пам'яті – три машинних цикли і десять тактів.

					ДП 5.05010201 435 01 ПЗ	Арк.
						40
Змін.	Арк.	№ докум.	Підпис	Дата		

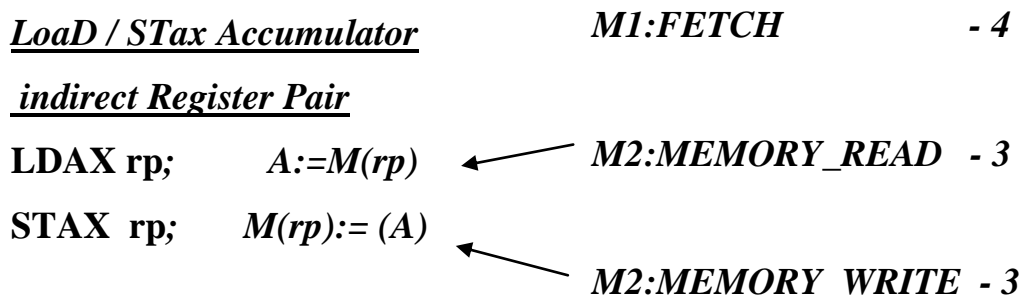


Відрізняються команди типом четвертого машинного циклу. Командний цикл обох має тринадцять тактів.

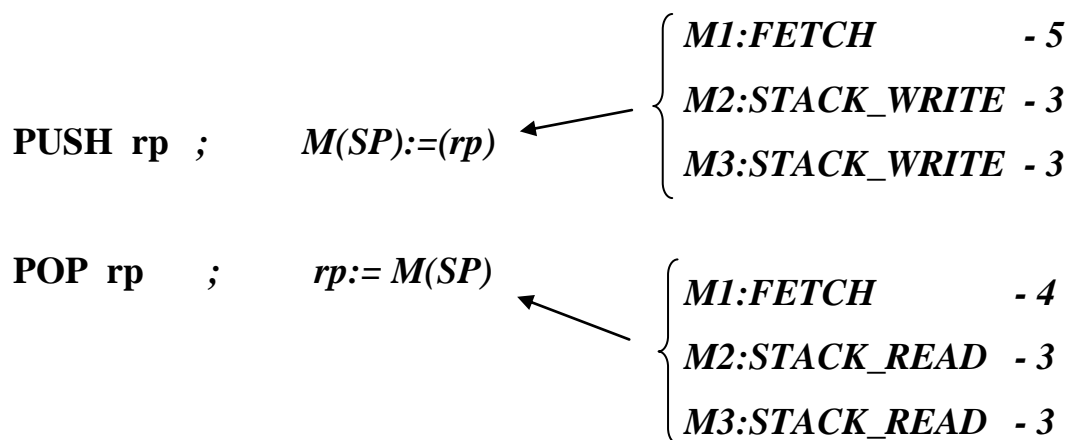
Команди формату 1.4 (*Load/Store register pair HL Direct ADDRESS*) загрузки/збереження регістрової пари за прямою адресою подібні попередній парі команд своєю байтністю і часом їх читанням із програмної пам'яті та часом їх виконання в шістнадцять тактів. Різниця в типі машинних циклів *M4, M5*.



Простота команд формату 1.5 безсумнівна тому, що виконуються подібні переміщення байтів між акумулятором і непрямо адресуєму через регістрову пару *BC, DE* за два машинних цикла та сім тактів.



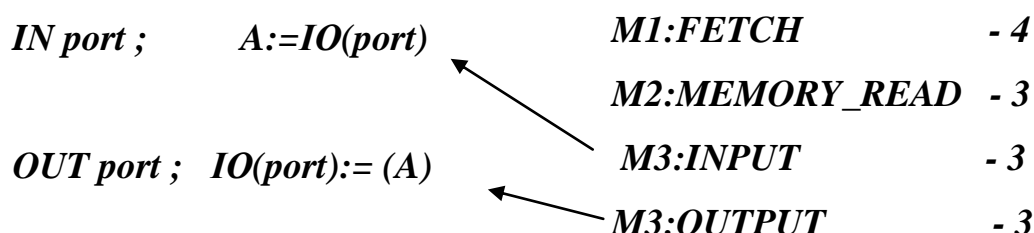
Стекові команди забезпечують передачу слів між регістровою парою та пам'яттю. Це є вигідним при перериваннях та при викликах процедур, оскільки теку-чий регістровий статус можна тимчасово зберігати в стеку.



При виконанні завантаження в стек на циклі ***M1:FETCH*** та на стані ***T5*** показ-ник стеку ***SP*** зменшується на одиницю з метою формування нової вершини. Після ви-гризки байту з вершини стеку показник ***SP*** збільшується на одиницю на текучих ма-шинних циклах ***STACK_READ***. Все це передбачено архітектурно і виконується при-ховано.

Команда ***SPHL*** з алгоритмом завантаження **$SP:=(HL)$** будує нову вершину сте-ку за адресою зі стану регістрової пари ***HL***. Обмін слів між парами регістрів ***DE*** і ***HL*** виконується по однобайтній одноцикловій чотирьохтактній команді ***XCHG*** (***eXCHanGe***). Є ще одна команда обміну словами між вершиною стеку та регістровою ***HL*** парою ***XTHL*** (***eXchange Top of stack and register pair HL***), на що процесор витра-чає п'ять машинних циклу з числом тактів вісімнадцять.

Команди вводу-виводу трьох циклові, десятитактні передають байти між акуму-лятором процесора та портами, які прямо адресовані із тіла команди та розміщені в ізольованому просторі вводу виводу.



1.4.2 Команди логічної обробки

Група логічних команд має базовий набір інструкцій для реалізації кон'юнкції **ANA** (**ANd Accumulator**), диз'юнкції **ORA** (**OR Accumulator**), суми по модулю два **XRA** (**eXluseve oR Accumulator**) і заперечення **CMA** (**CoMplement Accumulator**). Остання унітарна команда порозумово адресує акумулятор і не впливає на прапори. Попередні три адресують другий операнд в якості доступного регістру чи комірки пам'яті даних, а перший операнд знаходиться в акумуляторі. Важливо пам'ятати, що ці команди впливають лише на три прапори **S**, **Z**, **P**. Прапори перенесень **CY**, **AC** не зачіпляються. Склад логічних команд у скороченому визі зображено унизу.

2.1 ANA/ORA/XRA/CMP r/M 1:1:4/2:7

2.2 ANI/ORI/XRI/CPI data 2:2:7

2.3 RLC/RRC/RAL/RAR 1:1:4

2.4 CMA/STC/CMC 1:1:4

Команда порівняння **CMP** (**CoMPare**), або ще непорушаємого віднімання дуже схожа на віднімання в питанні впливу на усі п'ять прапорів **S**, **Z**, **P**, **CY**, **AC**, але без зміни зменшуємого.

ANA	}	<i>r/M</i> →	<u>A:=(A) oper (r) / M(HL)</u>	
ORA				
XRA			M1:FETCH	- 4
CMP			M2:MEMORY_READ	- 3

При адресації регістра команди за часом виконання одноциклові, а при адресації пам'яті – двохциклові і семитактні. Логічні обчислення виконуються в обладнанні АЛП мікропроцесора з блокуванням розповсюдження перенесень та активізацією відповідної функції.

					ДП 5.05010201 435 01 ПЗ	Арк.
						43
Змін.	Арк.	№ докум.	Підпис	Дата		

Наступний блок двохбайтних двохциклових команд за часом виконання в сім тактів в якості першого операнду підрозумівається акумулятор, а другий операнд є безпосередній ***I (Immediate)*** із тіла команди. Вплив на прапори подібний попереднім відповідним командам.

<i>ANI</i>	}	<i>data</i>	<u><i>A:=(A) oper data</i></u>
<i>ORI</i>			
<i>XRI</i>			<i>M1:FETCH - 4</i>
<i>CMI</i>			<i>M2:MEMORY_READ - 3</i>

Ко-

манди зсуву поділені на дві підгрупи – циклічного та розширеного циклічного (арифметичного). Ці команди є безоперандами (завжди зсув виконується над складом акумулятора) по зміщенню байту вправо чи вліво на один розряд за час їх виконання. Командний цикл складає один машинний ***M1:FETCH - 4*** з числом чотири такти.

<i>RLC –Rotate Left Cycle</i>	<i>A:=(A)*2 or (A7 div 128); CY:=(A7)</i>
<i>RRC – Rotate Right Cycle</i>	<i>A:=((A) div 2) or (A0)*128); CY:=(A0)</i>
<i>RAL – Rotate Arithmetic Left</i>	<i>A:=(A)*2 or ((CY) div 256); CY:=(A7)</i>
<i>RAR – Rotate Arithmetic Right</i>	<i>A:=((A) div 2) or (CY)*256); CY:=(A0)</i>

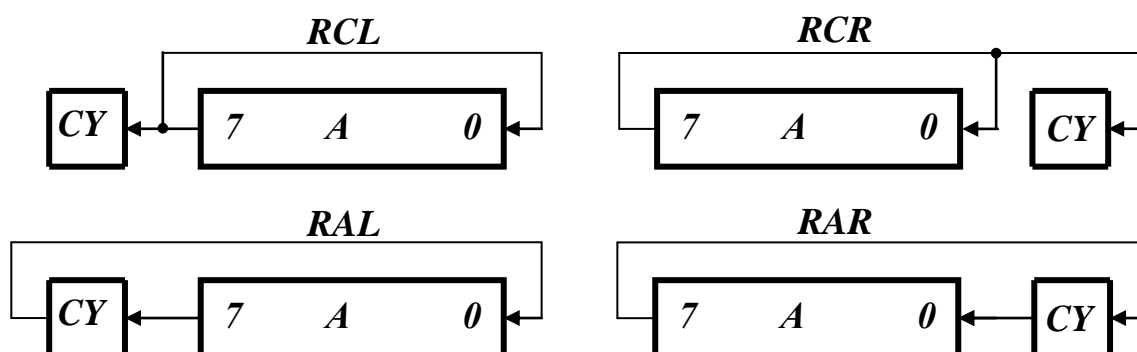


Рисунок 17 – Схема виконання в мікропроцесорі команд зсуву

Зсув у процесорі виконується в один такт за рахунок двохфазної синхронізації при тактуванні процесу запису. Друга ступінь пам'яті акумулятора – регістр **ТА**. Він має дуже розвинуті вхідні лінії запису, які є виходами мультиплексора, розміщеного між акумулятором і його буфером **ТА**. Скошена на один розряд між регістрова передача від акумулятора в його буфер по сигналу **F1** забезпечує буферизацію зсуву. Пряма передача від буфера акумулятора в акумулятор по сигналу **F2** четвертого такту **T4** машинного циклу **M1** є завершальною по зсуву. На малюнку унизу схематично висвітлено роботу команди **RCL** циклічного зсуву уліво. Стан прапора **CY** визначається станом сьомого біту акумулятора до виконання команди. Це значення також завантажується в нульовий розряд акумулятора, що є свідотством кільцевого зсуву уліво.

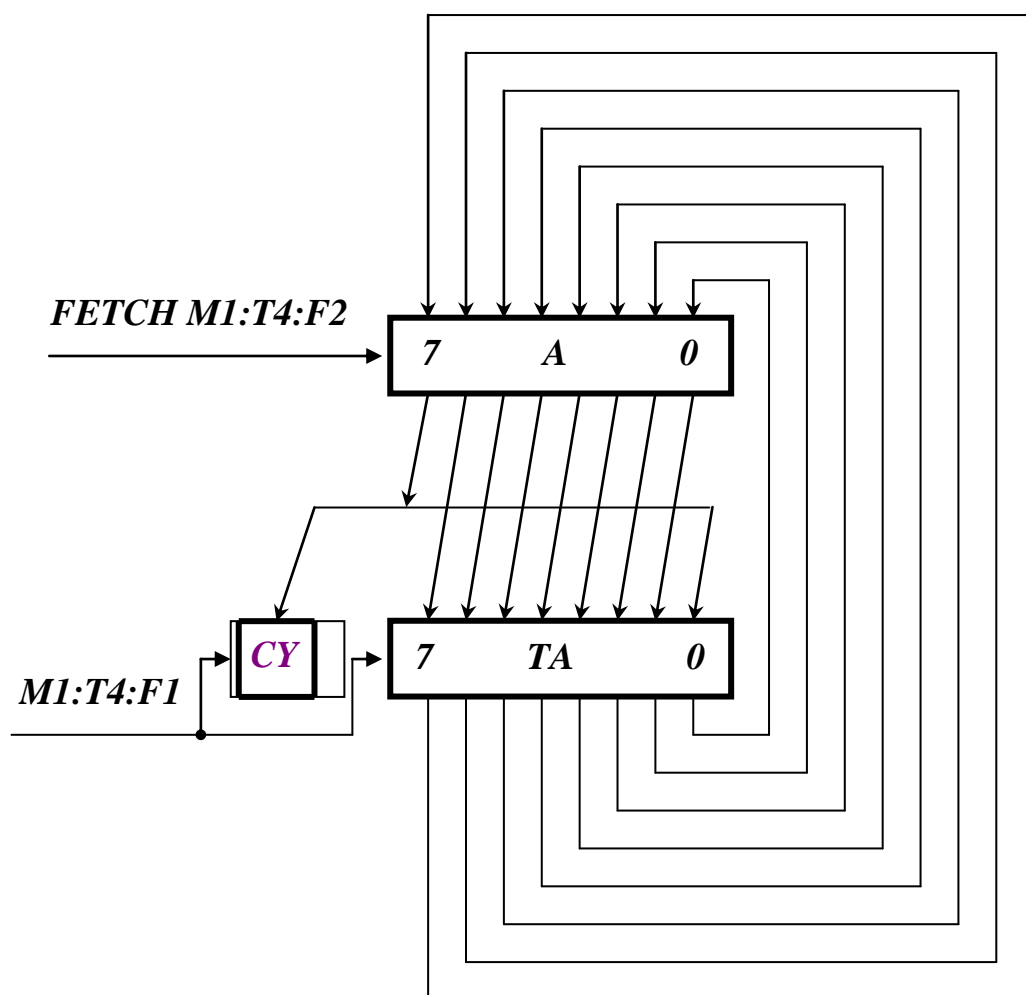


Рисунок 18 – Алгоритм циклічного зсуву уліво на один розряд

Виконання арифметичного зсуву управо **RAR** передбачає використання буферного розряду **BC** з метою урахування випианутого розряду для запису в прапор **CY**. Процес синхронізації запису даних в регістри при передачі спочатку скошеної, а потім прямої аналогічен попередньому.

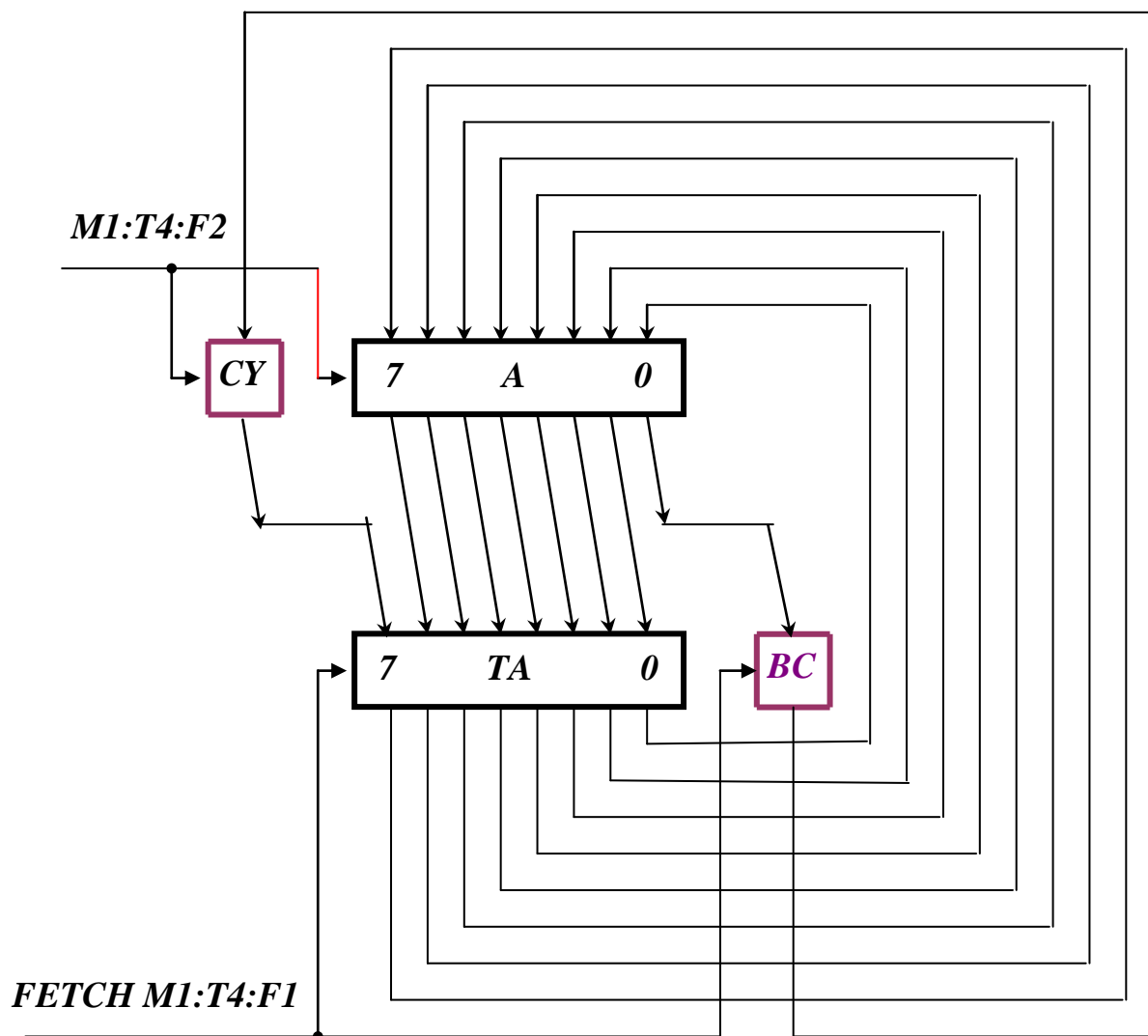


Рисунок 19– Алгоритм арифметичного зсуву управо на один розряд

Команда заперечення **CMA** спонукає виконанню функції $A := \text{not}(A)$, яка потрібна в обчисленнях по перетворенню кодів. Дві команди **STC** (*SeT Carry*), **CMC** (*CoMplement Carry*) по доступу до прапора **CY** забезпечують установку в одиницю $CY := 1$ та інверсію стану прапору перенесення $CY := \text{not}(CY)$, що є дуже важливим в обчисленнях. Поєднавши між собою логічні команди в програмному модулі, можна реалізувати скільки угодно складні алгоритми.

					ДП 5.05010201 435 01 ПЗ	Арк.
						46
Змін.	Арк.	№ докум.	Підпис	Дата		

1.4.3 Команди арифметичної обробки

Команди арифметичного обчислення спираються на виконання мікрооперації додавання, що спричинено самою архітектурою арифметичного пристрою. При відніманні автоматично виконуються відповідні пересилки між регістрами з фіксацією об'єктного коду від'ємника на входах суматора та генерацією вхідного переносу і додання зі зменшуваним. Така дія виконується за один машинний цикл. Формат, спосіб адресації, час виконання арифметичних команд співпадає з цими ознаками логічних команд. Арифметичні команди впливають на всі прапори мікропроцесора.

3.1 ADD/ADC/SUB/SBB <i>r/M</i>	1:1:4/2:7
3.2 ADI/ACI/SUI/SBI <i>data</i>	1:2:7
3.3 INC/DCR <i>r/M</i>	1:1:5/3:10
3.4 INX/DCX <i>rp</i>	1:1:5
3.5 DAA	1:1:4
3.6 DAD <i>rp</i>	1:3:10

Система арифметичних команд передбачає виконання додавання та віднімання без урахування і з врахуванням перенесень та позик відповідно, які відображені станом прапора **CY**. Базові чотири арифметичні команди виконуються за чотири або сім тактів головної синхронізації. Завжди результат поміщається на місце першого операнду, яким є акумулятор. Перший операнд порушується, другий лишається незмінним. Це є класична архітектура будови арифметико-логічного пристрою акумуляторного типу.

ADD	} <i>r/M</i>	$A := (A) + (r) / M(HL)$
ADC		$A := (A) + (r) / M(HL)$
SUB		$A := (A) - (r) / M(HL) + (CY)$
SBB		$A := (A) - (r) / M(HL) - (CY)$

Ураховуючи структуру АЛП на аркуші сторінки 21 посібника, виконання віднімання проводиться на четвертому такті першого машинного циклу.

SUB r ; SUstract M1:FETCH - 4

M1:T4:F1 T := (r) ; Timer register TA:= (A) ; Timer Accumulator

M1:T4:F2 (A):= (TA) + not(T) +1

<i>ADI</i>	}	<i>data</i>	<i>A:=(A) oper data ± 0 / (CY)</i>	
<i>ACI</i>				
<i>SUI</i>			<i>M1:FETCH - 4</i>	
<i>SBI</i>			<i>M2:MEMORY_READ - 3</i>	

При адресації безпосереднього операнду усі погодження лишаються не змінними. Змінюється тільки час виконання за рахунок циклу читання другого байту команди ***M2: MEMORY_READ*** та такт дії обчислення ***T3***.

Команди інкременту (***INCrement***) – декременту (***DeCRement***) складають підгрупу арифметичних команд. Корекція адресуемого командою об'єкту, яким є байт чи слово, виконується по різному. Це пов'язано з тим, що байти обробляються в арифметичному пристрої, а слова із регістрових пар обчислюються спеціальною схемою.

		<i><u>r/M :=(r/M) ± 1</u></i>	
<i>INC</i>	}	<i>r/M</i>	<i>M1:FETCH - 4</i>
<i>DCR</i>			<i>M2:MEMORY_READ - 3</i>
			<i>M3:MEMORY_WRITE - 3</i>

При адресації регістра команда одноциклова і п'ятитактна, а при адресації пам'яті вона трьохциклова та десятитактна. Команди змінюють лише чотири прапори ***S, Z, P, AC***. Прапор ***CY*** не змінюється по визначенням на то причинами.

Інші обставини виникають при програмній корекції слів із регістрових пар мікропроцесора. Обробка виконується схемою *INC/DEC*, про що раніше уже описувалося. Тому, обминаючи процесор в обчисленнях, усі п'ять прапорів не чіпляються ознаками і вони не змінюються.

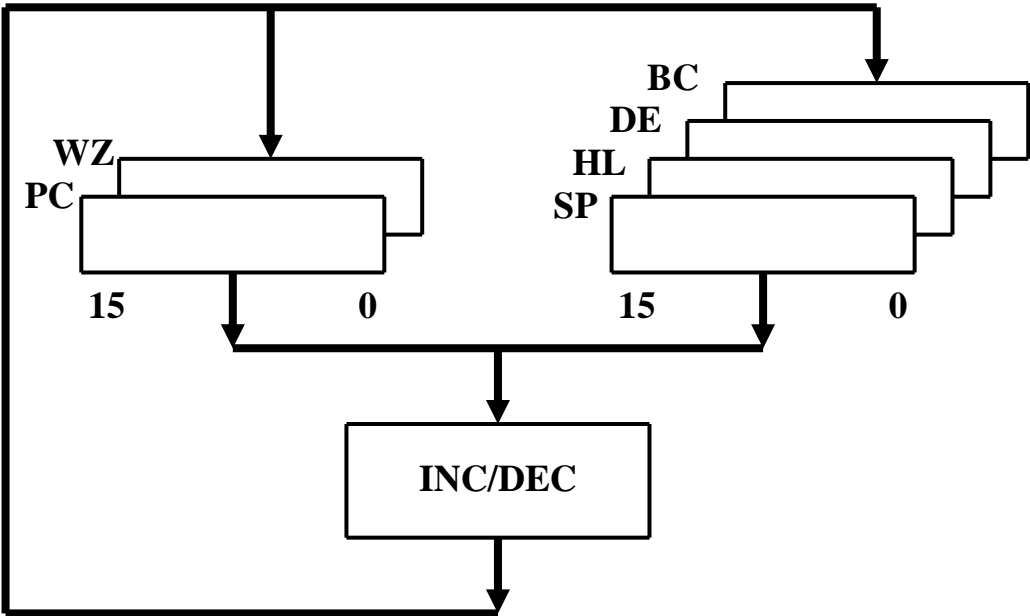


Рисунок 20 – Виконання інкременту-декременту регістрових пар

Команди *INX rp*, *DCX rp* виконуються за п'ять тактів і один машинний цикл. Вони адресують чотири регістрові пари *BC, DE, HL, SP*. Дві пари регістрів *WZ* і *PC*, які також потребують інкременту, обробляються схемою але не на програмному рівні, а на апаратному та приховано від користувача. В циклічних програмах функція програмного інкременту-декременту необхідна для будови програмного лічильника. Якщо по прапору ноля *Z* після корекції лічильника на базі регістрової пари виконувати передачу керування, то це буде помилкою. Для реалізації цієї складності необхідно іншим програмним способом тестувати на ноль лічильник.

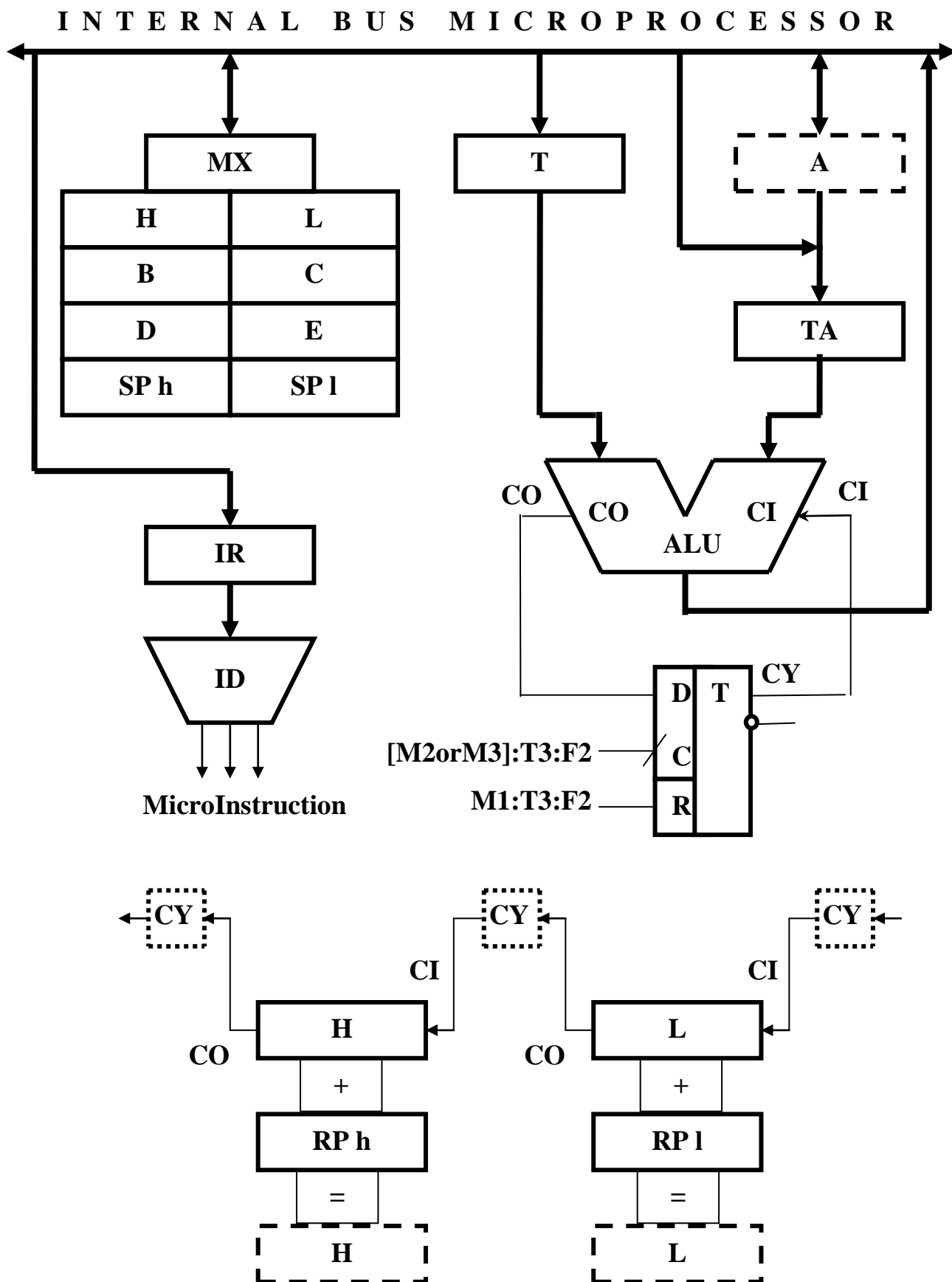


Рисунок 21 – Операційний пристрій при виконанні двійного додавання

Змін.	Арк.	№ докум.	Підпис	Дата

ДП 5.05010201 435 01 ПЗ

Арк.

50

І нарешті остання команда цієї групи – команда двійного додавання слів із регістрових пар **DAD rp** (**Double ADding Register Pair**). По цій команді виконується операція додавання

$$HL := (HL) + (rp).$$

В якості операндів є слова із відповідних регістрових пар (Треба ж було придумати додавання в восьми розрядному процесорі двохбайтових слів – примітка автора). Команда адресує другий операнд – одну із чотирьох регістрових пар, включаючи пару **HL**. Джерелом першого операнду та приймачем результату виступає пара регістрів **HL**. До речі, команда **DAD H** виконує додавання

$$HL := (HL) + (HL) = (HL) * 2,$$

що відповідає арифметичному зсуву управо на один розряд складу пари **HL**. Команда виконується за три цикли і десять тактів синхронізації

M1:FETCH - 4 ; **IR** := **M(PC)**

M2:MEMORY_READ - 3 ; **L** := (**L**) + (**rp_L**) + 0

M3:MEMORY_READ - 3 ; **H** := (**H**) + (**rp_H**) + (**CY**)

На циклах **M2:MEMORY_READ**, **M3:MEMORY_READ** процесор не формує сигнал **DBIN** тому пам'ять не читається. На цих циклах, як видно з малюнку, виконується побайтно додавання з урахуванням міжбайтового перенесення. При додаванні молодших байтів вхідне перенесення **CI** = 0 за рахунок попереднього скиду в ноль тригера. Після обробки старших байтів сумісно з формуванням результату виставляється прапор **CY**. На другі чотири прапори команда не впливає.

Неможливість покомандного множення і ділення спричинює будову достатньо складних алгоритмів, реалізація яких частіш за все потребує напису циклічних програм. Цей недолі виправлено в сучасних процесорах, які можуть виконувати названі арифметичні дії даже над знаковими операндами.

					ДП 5.05010201 435 01 ПЗ	Арк.
						51
Змін.	Арк.	№ докум.	Підпис	Дата		

1.4.4 Команди передачі управління

Команди цієї групи змінюють хід виконання команд програми, порушуючи тим самим натуральний порядок їх виконання. Перелік команд галуження не великий .

4.1 PCHL; **1:1:5**

4.2 JMP addr ; **3:3:10**

4.3 J cc addr; **3:3:10**

4.4 CALL addr; **3:5:17**

4.5 C cc addr; **3:3:11/5:17**

4.6 RET; **1:3:10**

4.7 R cc; **1:1:5/3:11**

4.8 RST n ; **1:3:11**

Сама проста команда завантаження $PC := (HL)$ з мнемонікою **PCHL** та за часом виконання п'ять тактів. Безумовний перехід за прямою адресою виконується трьохбайтною командою **JMP addr**. Алгоритм роботи команди не складний.

M1:FETCH - 4 ; $IR := M(PC)$

M2:MEMORY_READ - 3 ; $Z := Addr_L$

На **M3:MEMORY_READ** - 3 ; $W := Addr_H$ $PC := WZ$)

кожному

із циклів такту **T2** виконується дія $PC := (PC) + 1$. Після завантаження пари **WZ** адресою на такті **M3 :T3** виконується передача $PC := (WZ)$.

Важливе місце займає команда прямого умовного переходу **J cc addr**, яка тестує (перевіряє) стан одного із чотирьох прапорів та їх інверсію. Всього є вісім умовних команд з відповідним суфіксом коду. Порівнюючі суфікси та стани відповідних прапорів мікропроцесора є істиною і умовою переходу на другу адресу.

C Carry **CY=1;** **NC Not Carry** **CY=0**

Z Zero **Z=1;** **NZ Not Zero** **Z=0**

M Minus **S=1;** **P Positive** **S=0**

PE Parity Even **P=1;** **PO Parity Odd** **P=0**

					ДП 5.05010201 435 01 ПЗ	Арк.
						52
Змін.	Арк.	№ докум.	Підпис	Дата		

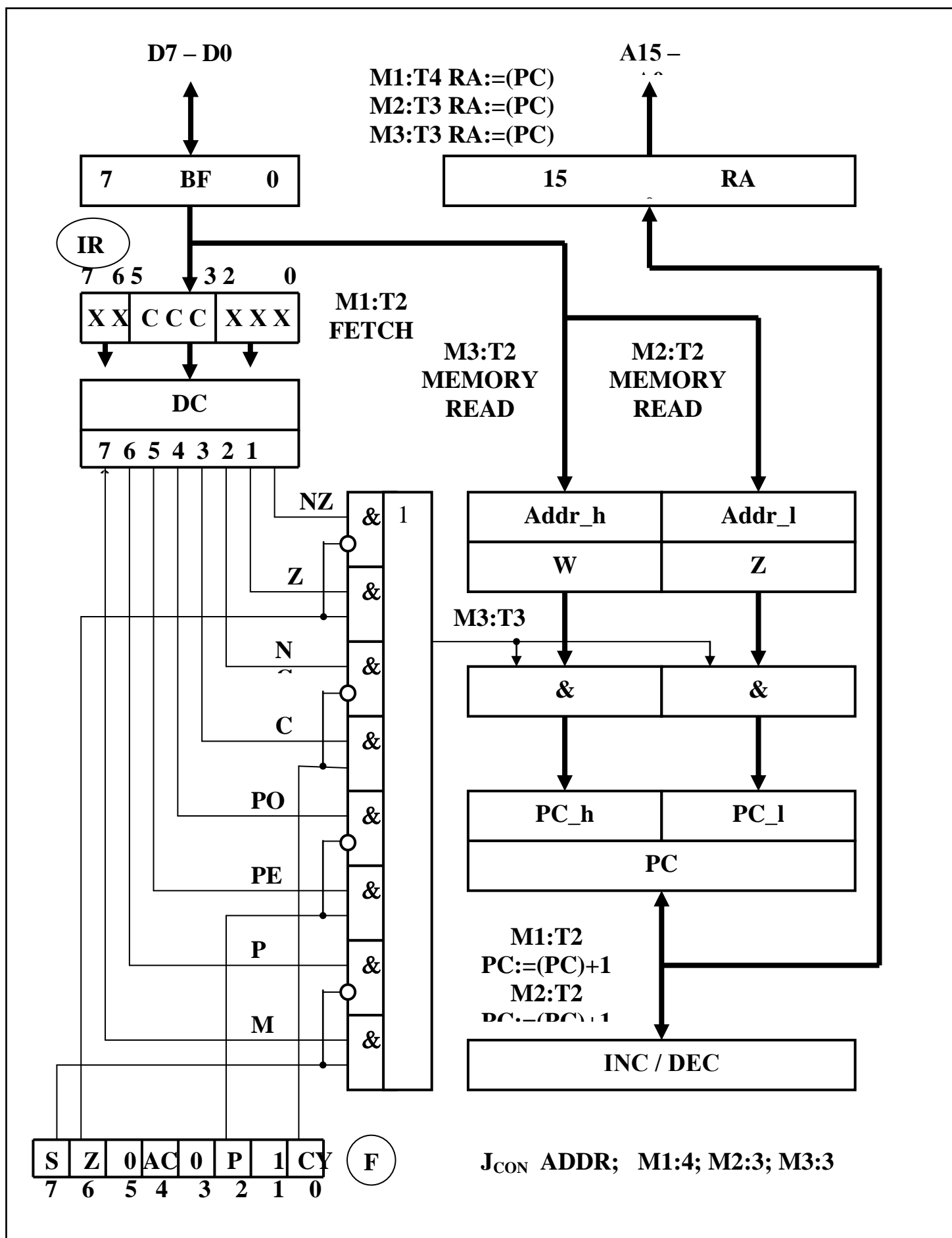


Рисунок 22 - Виконання в процесорі команди умовного переходу

Команда умовного переходу, як це показано на малюнку, виконується усього за десять тактів, але дії команди порівняно складні. На першому циклі другого такту ***M1:T2:FETCH*** в реєстр команд мікропроцесора ***IR*** завантажується код команди, три біти якого ***CCC (Condition)*** несуть інформацію про перевіряєму умову, яка відображена прапорами. Умова декодується дешифратором, який керує логічним формувачем вибору тестуємого прапору реєстру ***F*** мікропроцесора. У свою чергу, на останньому такті останнього машинного циклу ***M3:T3*** формувач виробляє високий чи низький рівень сигналу для передачі чи ні відповідно байтів ***Addr_h, Addr_l*** від реєстрів ***W, Z*** програмному лічильнику ***PC_h, PC_l***.

На тактах ***T2 [M2 or M3] :T2:MEMORY_READ*** другого та третього машинних циклів із програмної пам'яті завантажується в пару реєстрів ***WZ*** адреса галуження. На усіх трьох тактах програмний лічильник збільшується на плюс одиницю ***M1 or M2 or M3 :T2 PC:=(PC) +1***. Тому реєстр адреса ***RA***, приймаючи слова від програмного лічильника, спричинює звернення процесора до програмної пам'яті. На момент завершення командного циклу в сформується наступна адреса програмної пам'яті, а в реєстровій парі ***WZ*** – адреса галуження. (Треба ж, такі складності, а розв'язані просто ! Можливо і в суспільстві щось подібне присутнє ?... Примітка автора).

При будові алгоритмів та програмуванні виникають складнощі, пов'язані з визначенням умов порівнянь чисел в процесорі, критерії яких відображені прапорами мікропроцесора. Усього є шість таких критеріїв по три взаємо інверсних пари : рівно - нерівно; більше – менше чи нерівно; менше – більше чи нерівно. Операнди програмно можуть при їх порівнянні сприйматися знаковими чи беззнаковими числами. Операція порівняння схоже на віднімання в плані формування прапорів мікропроцесора. Тому, залежно від значень модулів чисел для беззнакових величин, прапори будуть мати відповідні величини.

Для визначення критеріїв порівнянь беззнакових чисел, що показано унизу, бажано користуватися інструментами законів алгебри логіки. Визначені прапори після порівнянь занесені в клітини мінімізуючої карти Карно, після чого проведено склеювання і визначені критерії умов порівнянь.

					<i>ДП 5.05010201 435 01 ПЗ</i>	Арк.
						54
Змін.	Арк.	№ докум.	Підпис	Дата		

E: If $x-y = 0$ then $S\ CY\ Z = 001$; Equate

A: If $x-y > 0$ then $S\ CY\ Z = 000$; Above

B: If $x-y < 0$ then $S\ CY\ Z = 110$; Below

E A B

		<i>CY Z</i>			
		<i>00</i>	<i>01</i>	<i>11</i>	<i>10</i>
<i>S</i>	<i>0</i>	<i>A</i>	<i>E</i>	<i>X</i>	<i>X</i>
	<i>1</i>	<i>X</i>	<i>X</i>	<i>B</i>	<i>X</i>

E = Z ; Equate

A = $\overline{CY} \ \& \ \overline{Z}$; Above

B = CY ; Below

NE = \overline{Z} ; Not Equate

NA = $CY \vee Z$; Not Above

NB = \overline{CY} ; Not Below

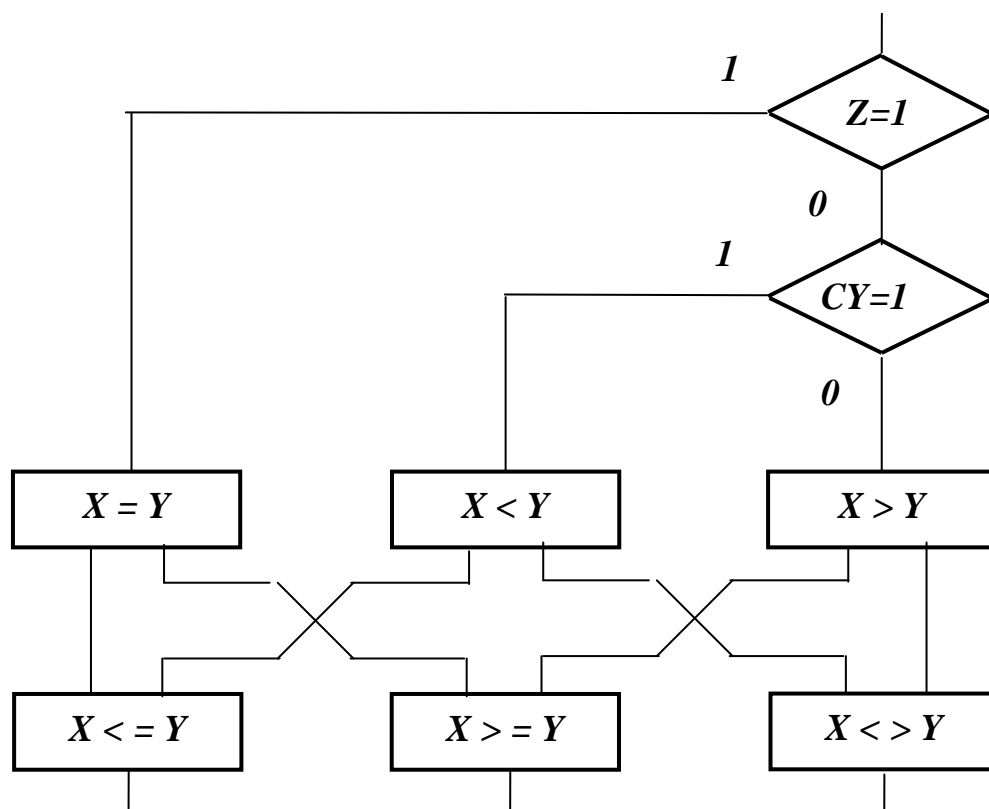


Рисунок 23 – Порівняння беззнакових чисел та умови критеріїв через прапори

Виклики процедур, як втім і переходи, поділяються на безумовні і умовні переходи. Команда безумовного виклику **CALL addr** має три байти, другий і третій якої складають пряму адресу програмної пам'яті, де знаходиться викликаєма процедура. Для читання з пам'яті команди в процесор потребується одинадцять тактів. Перший машинний цикл **FETCH** п'ятиактний, останній такт необхідний для корекції на мінус один показника стеку (дивись унизу малюнки). Два трьохтактні цикли **MEMORY_READ** забезпечують завантаження з пам'яті програм в регістрову **WZ** пару адреси виклику процедури. Кожен раз при зверненні процесора до програмної пам'яті склад програмного лічильника автоматично збільшувався на одиницю (**PC**) + **1**, що в кінцевому результаті призвело до формування адреси наступної команди.

Четвертим та п'ятим циклами є запис на вершину стеку **STACK_WRITE** складу програмного лічильника. Запис виконується в три такти – спочатку за старшою адресою з показника стеку завантажується старша частина **PC_H**, а потім молодша **PC_L**. Перед завантаженням склад показника автоматично зменшується на одиницю (**SP**) – **1**. Закінчується виконання команди пересилкою на останньому такті командного циклу слова із регістрової пари **WZ**, яким є адреса виклику, в програмний лічильник.

Таким чином, команда безумовного виклику спричинює в процесорі читання з програмної пам'яті трьох байтів – коду команди та двох байтів полуадрес – і запис в стек фізичної пам'яті полуадрес повернення в основну програму. При напису програм необхідно пам'ятати, що процедура завжди повинна мати у своєму тілі команду повернення. Якщо ця умова буде не виконана, то виникне невизначена ситуація.

На двох малюнках автором показані усі дії процесора не лише на машинних циклах, а і на кожному такті циклів. Крім того, досить оригінально відмічені шляхи передачі між фізичною пам'яттю та процесором адреси, команд і даних. Таке висвітлення складних питань будови та передачі інформації в мікропроцесорних пристроях дуже корисне при вивченні класичної структури системного стеку і системи переривань.

					ДП 5.05010201 435 01 ПЗ	Арк.
						56
Змін.	Арк.	№ докум.	Підпис	Дата		

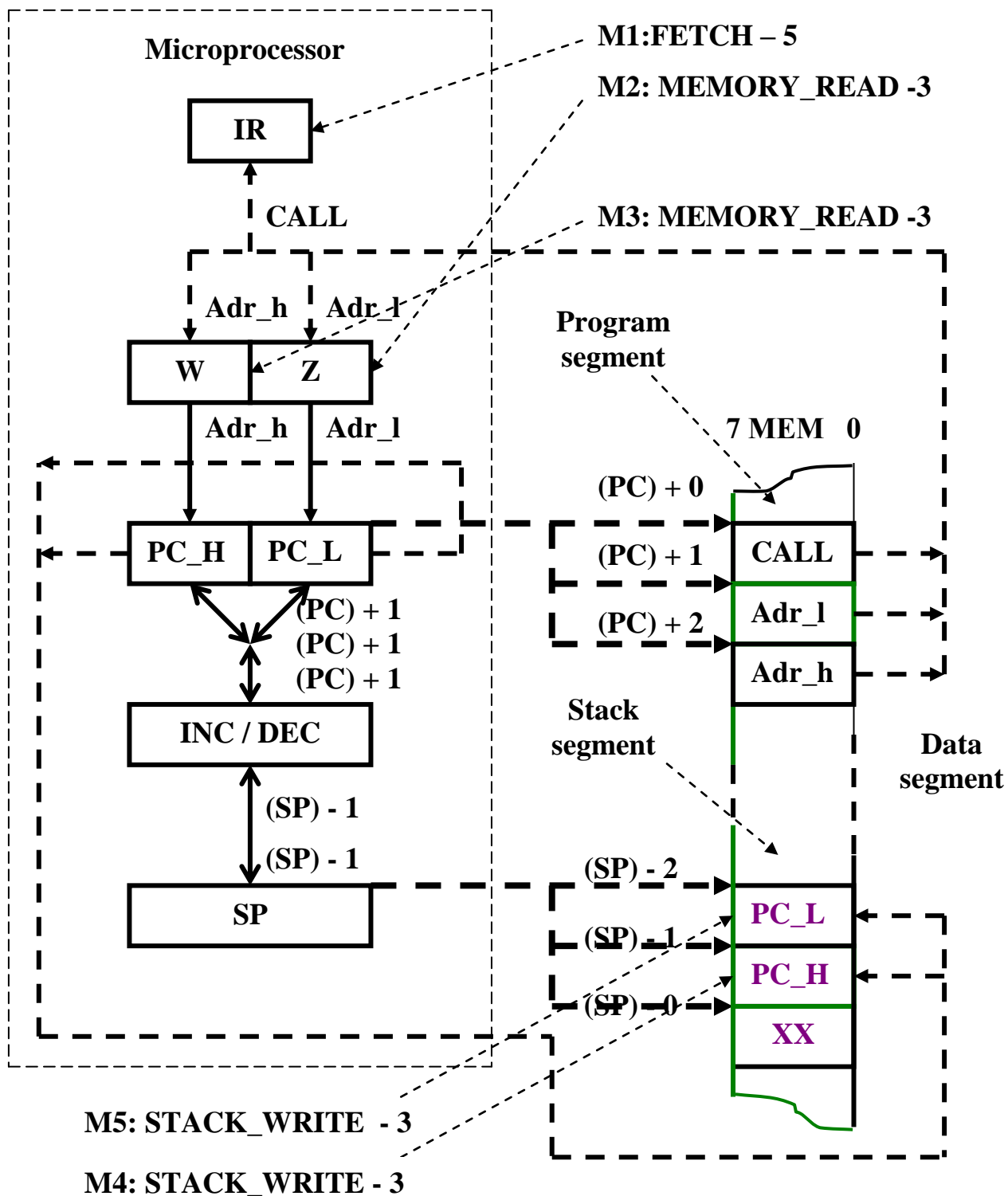


Рисунок 24– Обмін з пам'яттю при виконанні команди виклику

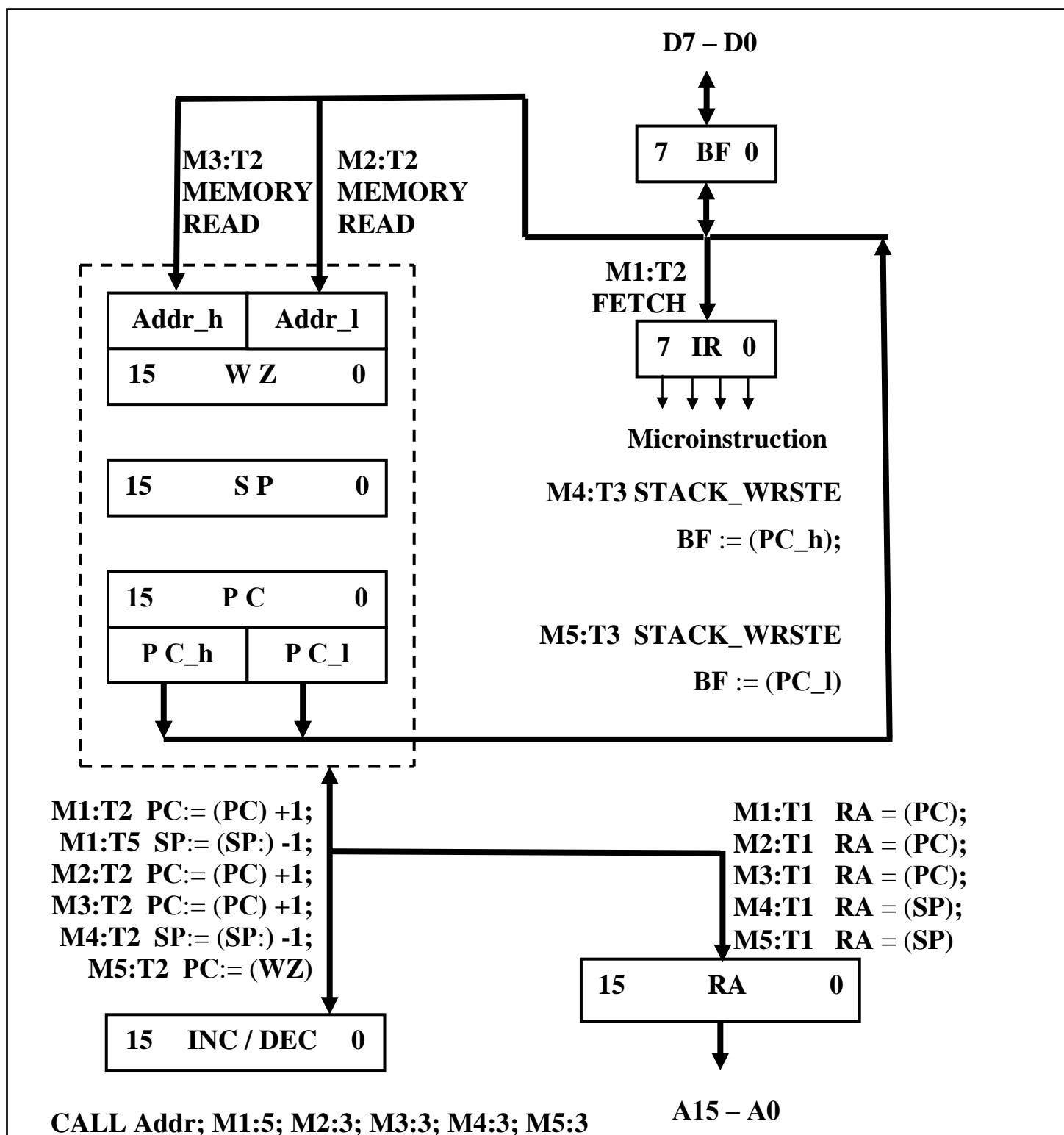


Рисунок 25 - Виконання в процесорі команди виклику процедури

Формат команди умовного виклику *Ccc addr* багато у чому схожий на безумовний та команду умовного переходу. Існує вісім відповідних команд умовного виклику процедури. Команда перевіряє одне із восьми умов, відображених в мікропроцесорі чотирма прапорами та їх інверсією, читає із програмної пам'яті адресу галуження, формує адресу наступної команди, при результаті істини тестування завантажує в стек адресу наступної команди а в програмний лічильник завантажує адресу виклику. При хибному результаті тестування командний цикл закінчується. Таким чином, командний цикл є величиною змінною (одинадцять або сімнадцять тактів) і залежи від результату апаратного внутрішнього тестування окремих ознак.

M1:FETCH - 5 ; ***IR := M(PC)***
M2:MEMORY_READ - 3 ; ***Z:= Addr_L***
M3:MEMORY_READ - 3 ; ***W:= Addr_H***
M4: STACK_WRITE - 3 ; ***M[(SP)-1] := ((PC)+3)_H***
M5:STACK_WRITE - 3 ; ***M[(SP)-2] := ((PC)+3)_L***

Скорочений алгоритм виконання команди свідчить, що внутрішня схема інкременту-декременту обслуговує не тільки програмний лічильник *PC* але і показник стеку *SP* . Регістрова пара *WZ* буферизує отриману адресу галуженням з метою забезпечення при умові істини тестування завантаження адреси в програмний лічильник.

Команда повернення із процедури ***RET (RETuun)*** досить проста в роботі. Однобайтний код команди після завантаження в регістр команд мікропроцесора на першому чотирьохтактному циклі спричинює виникненню послідовних двох циклів читання стеку, на яких адреса з вершини стеку завантажується в програмний лічильник. Спочатку завантажується молодша частина *PC_L*, а потім старша *PC_h*. Кожен раз при цьому склад показника стеку *SP* збільшується на одиницю з метою повернення його вершини в початковий стан, яки був порушений раніше викликом процедури . Командний цикл триває десять такти синхронізації.

					ДП 5.05010201 435 01 ПЗ	Арк.
						59
Змін.	Арк.	№ докум.	Підпис	Дата		

M1:FETCH - 4 ; IR := M(PC) (PC) := (PC)+1

M2: STACK_READ - 3 ; PC_l := M[(SP)] SP:=(SP)+1

M3:STACK_READ - 3 ; PC_h := M[(SP)] SP:=(SP)+1

Команда умовного виходу із процедури (по логіці мислення їх вісім) теж однокбайтна, але її командний цикл змінний і може бути рівним одному або трьом. Процесор стандартним способом тестує ознаку. Якщо умова тестування хибна, то на першому циклі такту ***T5*** склад ***PC*** збільшується на одиницю і на цьому циклі команди закінчується. При істинній умові тестування командний цикл збільшується на два машинних цикли по три такти кожен читання стеку.

M2: STACK_READ - 3 ; PC_l := M[(SP)] SP:=(SP)+1

M3:STACK_READ - 3 ; PC_h := M[(SP)] SP:=(SP)+1

І, нарешті, остання команда групи передачі керування – повторного пуску процесора за номером ***RST n (ReStart Number 0 .. 7)***. Ця команда стала відправною віточкою у розвитку команд програмного переривання сучасних мікропроцесорів родини ***Pentium***. Спільність полягає в передачі керування в уже передбачено відому область програмної пам'яті. Ця область заздалегідь завантажена прикладними програмами операційної системи – самотестування, обслуговування пристроїв передачі інформації та іншого обладнання.

M1:FETCH - 5 ; IR := M(PC) PC := (PC) +1 SP :=(SP) -1

M2: STACK_WRITE - 3 ; M[(SP)] := ((PC_H)

M3:STACK_WRITE - 3 ; M[(SP)] := ((PC_L) PC := (N)*8

На кожному такті кожного машинного циклу внутрішні регістри пристрої мікропроцесора сприймають відповідні дані, перетворюють їх та запам'ятовують для подальшого використання з метою виконання закладеного алгоритму роботи.

					<i>ДП 5.05010201 435 01 ПЗ</i>	Арк.
						60
Змін.	Арк.	№ докум.	Підпис	Дата		

Ареал комірок пам'яті із восьми блоків по вісім комірок у кожному загальною ємністю шістдесят чотири байти, які розміщені у початковій області фізичної пам'яті. Стандарт передачі управління на початкову область було доопрацьовано та реалізовано в сучасних мікросистемах для розміщення векторів переривань.

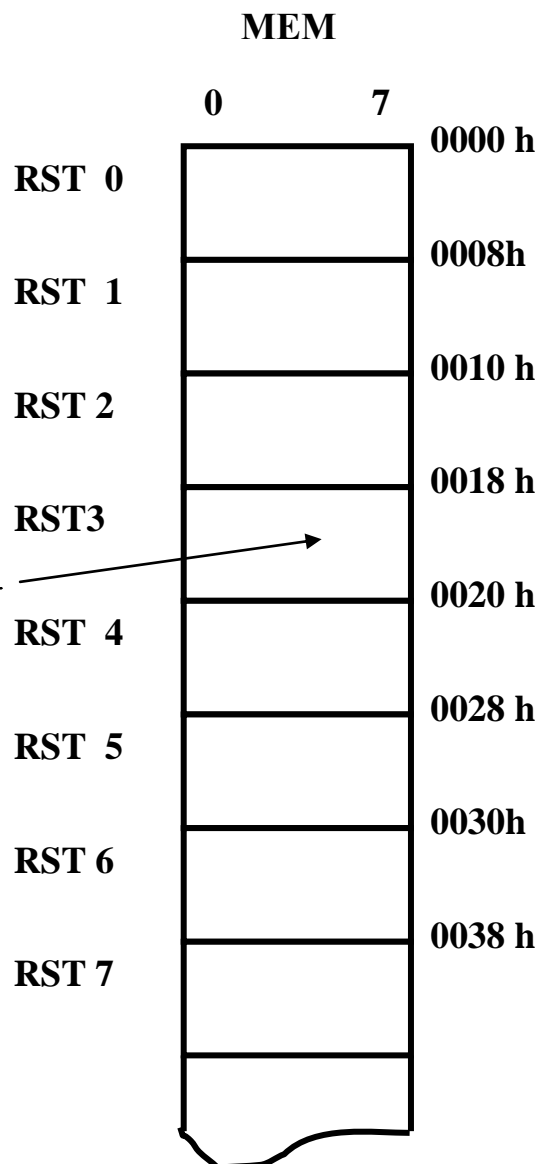


Рисунок 26 - Виконання початкової пам'яті командами перезапуску

Команда **RST n** може знаходитися в тілі програми, а також бути останньою. Усе залежить від версії операційної системи, доопрацьованого або самостійно складеного прикладного програмного забезпечення, складом периферійного обладнання мікропроцесорної системи, яке обслуговується за програмними перериваннями короткими процедурами.

Склад команд передачі управління (особливо умовних переходів) сумісно з групами арифметико – логічної обробки представляють, умовно кажучи, інтелектуально розумовий потенціал мікропроцесорної обчислювальної системи.

1.4.5 Команди управління мікропроцесором

Дві команди дозволу / заборони (*Enable / Disable Interrupt*) зовнішніх переривань *EI, DI* забезпечують установки скид вихідної лінії *INTE* дозволу зовнішніх переривань. Всього один машинний цикл та чотири такти існує командний цикл.

Цікавою є команда зупинки мікропроцесора *HLT*. Код команди **76 h** породжує мнемоніку *MOV M, M* команди пересилки пам'ять – пам'ять, чого робити не можуть сучасні процесори. Процесор, отримавши однобайтну команду зупинки, діє невизначено – з одного боку він хоче прочитати пам'ять, а з іншої сторони хоче цю комірку перетворити на приймач байту для запису.

M1:FETCH - 4 ; IR := M(PC) PC := (PC) +1

M2: HALT -3 ; Testing RESET or (INT & INTE)

Після першого циклу в *PC* сформується адреса наступної команди текучої програми. На машинному циклі зупинки *M2: HALT* процесор зі стану *T2* перемикається в стан зупинки і обчислення зупиняються. У цьому стані процесор може знаходитися до системного скиду по сигналу *RESET* або до переривання мікропроцесора, коли склад *PC* завантажиться в стек. Команда *RET* перериваючої процедури адресу повернення з вершини стеку завантажить в програмний лічильник і робота програми продовжиться. Цей механізм синхронізує тихохідні запити на переривання в процесор від периферійних пристроїв зі швидкодіючим процесором по виконанню програми.

Остання команда без операції *NOP (No Operation)* виконується за чотири такти *M1:FETCH* та лише змінює стан лічильника $PC := (PC) + 1$, що забезпечує перехід процесора на наступну команду. Використовується для резервування байту в робочій програмі або для термінової витримки часу в програмному циклі процедури керованої затримки. Команда *NOP* операційний пристрій мікропроцесора не чіпляє.

					ДП 5.05010201 435 01 ПЗ	Арк.
						62
Змін.	Арк.	№ докум.	Підпис	Дата		

1.5 Розробка програмного забезпечення тестування інтерфейсу пам'яті з головним мікропроцесором

Експлуатація мікропроцесорних систем за відсутності достатньо ефективних методів і засобів їх автоматизованої перевірки не можлива. Автоматизований контроль дозволяє поліпшити якість мікросистеми; підвищити достовірність вирішення завдань, надійність, готовність до роботи і безвідмовність; зменшити частоту помилок із-за збоїв; скоротити час відновлення нормальної працездатності системи при виникненні несправностей.

Існує два основні види контролю, що розрізняються способами отримання інформації помилково: апаратний і програмний контроль. Основний недолік апаратного контролю – значне кількісне збільшення загального об'єму апаратури мікросистеми і, отже, складнощі. Його гідність – мінімальний час контролю.

Програмні методи контролю можуть застосовуватися в процесі рішення задачі, вони не вимагають додаткового устаткування, але істотно знижують продуктивність комп'ютерів.

Зрозуміло, існує і комбінований програмно – апаратний контроль, що володіє достоїнствами обох методів контролю.

Особливий інтерес представляють спеціальні методи програмного контролю – діагностичні тести, які служать, для локалізації відмови і індикації місця, що відмовило, на пристрої відображення. Програмний контроль з використанням діагностичних тестів заснований на програмній доступності всіх вузлів мікропроцесорної системи, що дозволили центральному мікропроцесору видавати на ці вузли сигнали, що управляють, тестові набори і опитувати їх стан.

Програма самоконтролю повинна запускатися відразу після виконання подачі живлення і вироблення сигналу системного скидання. Програма послідовно повинна перевірити окремі частки мікросистеми.

Існує два підходи функціонального тестування напівпровідникової пам'яті, виявлення відмов і локалізації несправностей. У першому випадку пам'ять розглядається як цифровий автомат з відомим знаком функціонування.

					ДП 5.05010201 435 01 ПЗ	Арк.
						63
Змін.	Арк.	№ докум.	Підпис	Дата		

У другому випадку виконується блоковий підхід, коли функціональне тестування виконується за допомогою різних функціональних тестів, контролюючих окремі вузли або характеристики.

Пам'ять даних мікропроцесорної системи відноситься до двох напрямленої тому, вона дозволяє запис та читання даних. Використаний типовий стандартний інтерфейс модуля пам'яті дозволяє передавати при запису чи читанню між процесором і комірками слова, розрядність яких дорівнює байту. Байтна організація пам'яті значно спрощує алгоритми програмного тестування.

Процедура **RAMTST** забезпечує тестування інтерфейсу пам'яті даних ємкістю не більш 64К байт. Алгоритм тестування полягає в чотирикратному виклику процедури заповнення і порівняння **FICMP**. Перед кожним викликом програмно встановлюється тестовий код **00h, 0FFh, 0AAh, 55h**. Процедура **FILCMP** завантажує тестове значення у всі елементи тестованої пам'яті, а потім читається кожен осередок. Результат прочитування порівнюється тестовим значенням. В разі неспівпадання код прапор **СУ** встановлюється в одиницю, фіксується адреса “винуватою” осередки і здійснюється негайне повернення в основну програму тестування **RAMST**.

Основна програма аналізує ознаку помилки, відбиту одиничним станом прапора **СУ**. Якщо помилка в пам'яті є відбувається вихід з програми тестування з передачею управління в операційну систему, в результаті чого переривається виконання текучої програми.

Якщо помилка не виявлена, знову вся область пам'яті тестується кодом зі змінюючоюся одиницею по всіх розрядах. Кожен осередок (екомірка) восьмикратно модифікується. Ознака помилки роботи осередку також відображується прапорцем **СУ** шляхом установки в одиницю.. Після такої модифікації в кожен осередок завантажується код **00h** і відбувається перехід на черговий осередок до повного “перегляду” всій області пам'яті. Безпомилкова робота пам'яті після тестування характеризується нульовим станом прапора **СУ**.

					ДП 5.05010201 435 01 ПЗ	Арк.
						64
Змін.	Арк.	№ докум.	Підпис	Дата		

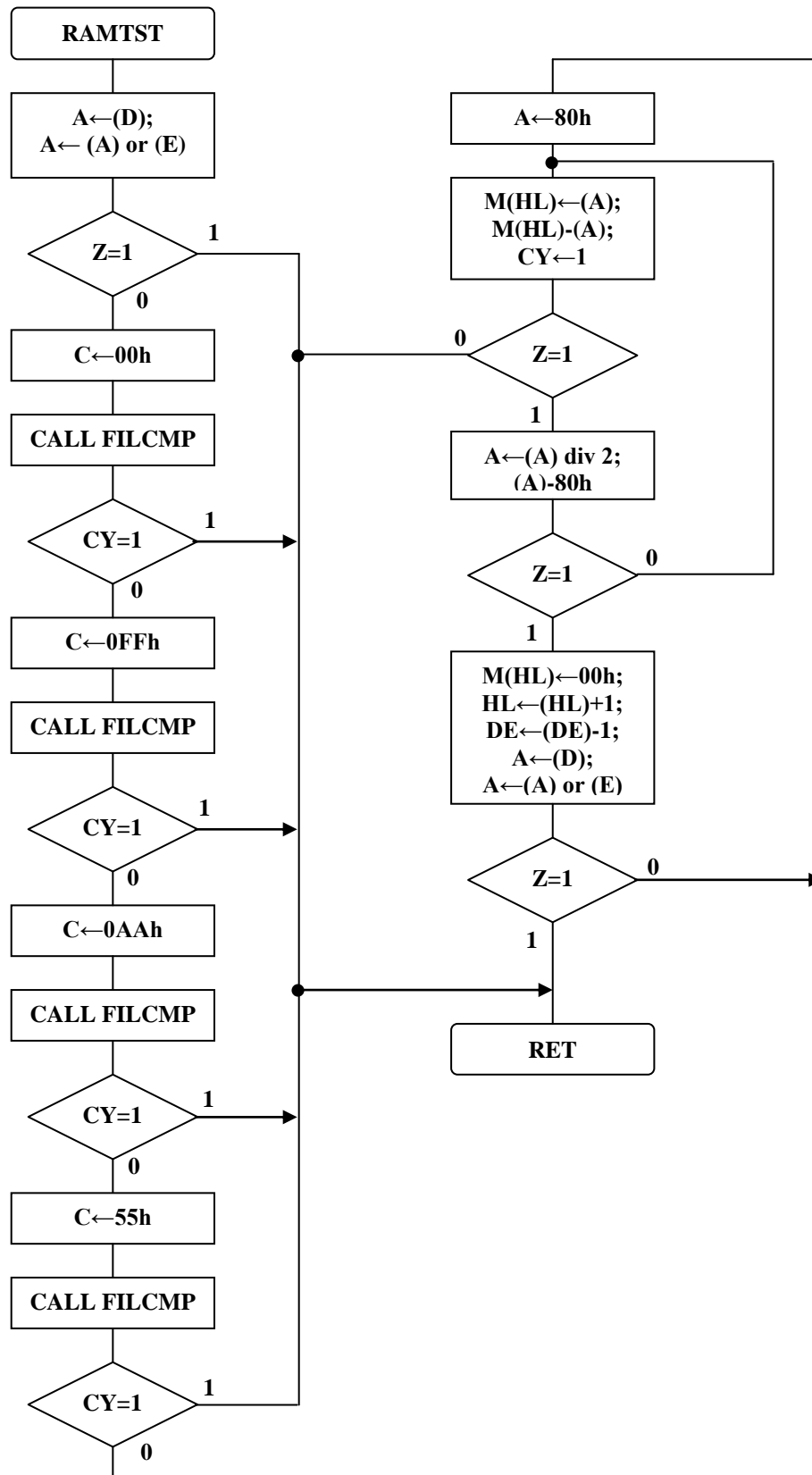


Рисунок 27- Структурна схема алгоритму тестування пам'яті даних

Змін.	Арк.	№ докум.	Підпис	Дата

ДП 5.05010201 435 01 ПЗ

Арк.

65

Лістинг 1 - Програма тестування пам'яті даних

```

RAMST:  MOV A, D          ; Нерівність нуля довжини
        ORA E
        RZ
        MVI C, 0          ; Тестування кодом 00h
        CALL FILCMP
        RC
        MVI C, 0FFh       ; Тестування кодом FFh
        CALL FILCMP
        RC
        MVI C, 0AAh       ; Тестування кодом AAh
        CALL FILCMP
        RC
        MVI C, 55h        ; Тестування кодом 55h
        CALL FILCMP
        RC

WKLP:   MVI A, 80h         ; Унітарний код 80h
WKLPI:  MOV M, A           ; Завантаження в поточний осередок коди
        MOV B, A           ; Збереження коди
        MOV A, M           ; Читання осередку
        CMP B              ; Порівняння з тестовим кодом
        MOV A, B           ; Відновлення коди
        STC                ; Відладка помилки
        RNZ
        RRC                ; Переміщення одиниці коди
        CPI 80h            ; Перевірка на вихід з циклу
        JNZ WRLIP
        MVI M, 0           ; Завантаження нуля в осередок
    
```

INX H ; Просування покажчика адреси
DCX D ; Зменшення довжини пам'яті
MOV A, D ; Перевірка виходу з циклу
ORA E
JNZ WKLP
RET

;Процедура заповнення і порівняння

FILCMP: PUSH H ; Збереження в стеку покажчика
PUSH D

FILLP: MOV M, C ; Завантаження тестованого коду в осередок
INX H ; Корекція покажчика
DCX D ; Зменшення довжини
MOV A, D ; Перевірка на нуль довжини
ORA E
JNZ FILLP

POP D ; Відновлення із стека
POP H

PUSH H ; Збереження в стеку покажчика
PUSH D

CMPLP: MOV A, M ; Читання поточного осередку
CMP C ; Порівняння з тестовим кодом
JNZ CMPPER
INX H ; Корекція покажчика адреси
DCX D ; Декремент на одиницю довжини
MOV A, D ; Перевірка на нуль довгі
ORA E
JNZ CMPLP ; Відновлення із стека
POP D

POP H

RET

CMPER: MOV A, C

POP D ; Відновлення із стека

POP B

STC ; Установка помилки

RET

					<i>ДП 5.05010201 435 01 ПЗ</i>	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		68