Лабораторна робота № 4

Тема: Система команд процесора I8086.

Мета: Ознайомитися із способами адресації операндів та основними групами команд процесора I8086.

Короткі теоретичні відомості

1. Способи адресації операндів у процесора I8086

Кожна команда процесора складається з двох частин: операційної і адресної. Операційна частина (код операції) вказує яку дію потрібно виконати. Адресна частина вказує, де знаходиться інформація і куди потрібно відправити результат. Адресна частина може бути відсутня (NOP, HALT).

Найчастіше використовують двоадресні команди. Вони вказують на дію, яку потрібно виконати над першим і другим операндом. Двоадресні команди не мають окремої адреси результату. Результат розміщується на місці одного з операндів або в спеціальному регістрі – акумуляторі. Адресація процесора I8086 належить до типу «регістр – пам'ять». Це означає, що команди мають максимум два операнди і не допускають одночасної адресації двох комірок пам'яті. Загальна структура команди така:

КОП dst, src (де КОП. – код операції, dst – приймач даних, src – джерело даних).

Варіанти адресації операндів
Регістр – регістр
Регістр – пам'ять
Пам'ять – регістр
Регістр – константа
Пам'ять – константа

Процесор I8086 має кілька способів адресації операндів.

1. Регістрова адресація. Операнди – регістри процесора.	`mov ax, bx` `add ax, si`	Передати bx в ax Додати si до ax
2. Безпосередня адресація. Операнд (константа) міститься в команді	`mov ax, 2` `add al, 30h`	Занести 2 в ax Додати 30h до al
3. Пряма адресація. Адреса операнда міститься в команді	`mov ax, text1` `mov bx, es:0002`	Занести змінну text1 в ax
4. Непряма регістрова адресація. Адреса операнда знаходиться в одному з базових або індексних регістрів (BX, SI, DI)	`mov [si], cl` `add al, [bx]`	Занести cl в комірку, адреса якої знаходиться в si Додати до al вміст комірки на яку вказує bx
5. Базова адресація (база плюс зміщення). Адреса операнда є сумою вмісту регістра BX або BP і зміщення	`mov ax, 2[bp]` `add [bx+10], ax`	Занести в ax слово за адресою bp+2 Додати до шостого слова в масиві, на який вказує bx, вміст регістра ax
6. Індексна адресація (база плюс індекс) Адреса операнда є сумою вмісту регістра SI або DI і зміщення	`mov 6[si], ax` `add cx, [di+2]`	Передати ax в елемент масиву Додати елемент масиву до cx
7. Базова індексна адресація (база плюс індекс плюс зміщення). Адреса операнда є сумою вмісту базового регістра, індексного регістра (BX + SI, BX + DI, BP + SI и BP + DI) і зміщення	`mov ax, 2[bx][si]` `mov ax, [bx+si+2]`	Передати елемент масиву в ax

Основні групи команд

Система команд будь-якого процесора складається із кількох груп команд. Наведемо приклади команд основних груп команд процесора I8086.

1. Команди передачі даних

Команда	Дія	Приклади

MOV dst, src	Передає вміст джерела src в приймач dst	mov dx,cx
		mov bp, gamma
		mov [bx], 3450h

XCHG reg, mem/reg	Обмінює вміст пари регістрів або регістра і комірки пам'яті	xchg al, bl xchg ax, cx xchg cx, [bp]

XLAT	Дістає байт з комірки пам'яті із зміщенням (ефективною адресою) BX+AL і заносить в AL	mov bx, table
		mov al, adr
		xlat

LEA reg, mem	Завантажує ефективну адресу джерела в регістр	lea bp, alpha lea bx, [bx][si]
IN acc, port	Завантажує дані з порта в акумулятор	in al, 40h in al, dx
OUT port, acc	Виводить дані з акумулятора в порт	out 80h, al out dx, ax

2. Команди арифметичних операцій

ADD dst, src (mem/reg, mem/reg)	Додає операнди dst та src і розміщує результат в dst	add cx, dx
		add cx, 0ABDCh
		add ax, temp

ADC dst, src	Додає операнди dst, src та прапорець CF і розміщує результат в dst	adc al, 35h adc [di], ax
INC dst	Збільшує на 1 регістр або комірку пам'яті	inc [di] inc si

SUB dst, src	Віднімає src від dst і розміщує результат в dst	sub cx, bx
		sub cl, [bp+2]
		sub al, 10h

SBB dst, src	Віднімає src та позику CF від dst і розміщує результат в dst	sbb ax, bp sbb [di], 30h
DEC dst	Зменшує на 1 регістр або комірку пам'яті	dec [bp] dec al
NEG dst	Змінює знак операнда в доповняльному коді	neg bx
CMP dst, src	Порівнює операнди (dst – src)	cmp al, 0dh
MUL src	Множить беззнаковий операнд на акумулятор (AX:=AL* src або [DX–AX]:=AX* src	mul dl mul [si+15]
IMUL src	Множить знакові операнди (AX:=AL* src або [DX–AX]:=AX* src	imul dl imul alpha
DIV src	Ділить беззнакові операнди на акумулятор (AL:=AX/src або AX:=[DX–AX]/ src	div cl div [bp+20]
IDIV src	Ділить знакові операнди на акумулятор (AL:=AX/src або AX:=[DX–AX]/ src	idiv ch

3. Логічні операції

Логічні команди AND, TEST, OR, XOR впливають на прапорці. OF, CF переводяться в нульовий стан, SF, ZF, PF залежать від результату операції. AF не визначений.

Команда NOT не впливає на прапорці.

AND dst, src	*dst := (dct)* $\wedge$ *(src)*, логічне І, AND	and ax, 00001111b
TEST dst, src	*(dct)* $\wedge$ *(src)*, логічна перевірка	test [si], cx
OR dst, src	*dst := (dct)* $\vee$ *(src)*, логічне АБО, OR	or temp, dl
XOR dst, src	*dst := (dct)* $\oplus$ *(src)*, виключне АБО, XOR	xor ax, dx
NOT src	*src :=* $(\overline{src})$, інверсія, NOT	not ax

Нагадаємо, що логічні операції означають наступне:

X	Y	X AND Y	X OR Y	X XOR Y	NOT X
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

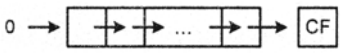
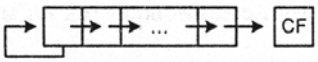
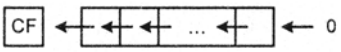
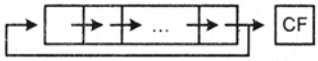
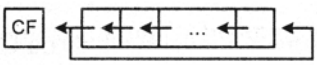
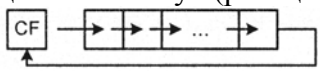
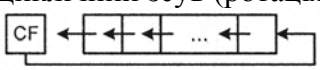
Приклади логічних операцій над даними:

	AND		OR		XOR			
X (дані)	00110111	37h	00000111	07h	00110111	37h	00110111	37h
Y (маска)	00001111	0Fh	00110000	30h	00001111	0Fh	00110111	37h
Результат	00000111	07h	00110111	37h	00111000	38h	00000000	00h

Логічні операції у мові асемблера використовують для керування окремими бітами даних. При цьому перший операнд називають даними, а другий маскою. Маска показує з якими бітами будуть виконуватися дії. Маска може мати активні "одиниці" або "нулі". Якщо необхідно деякі біти даних перевести в нуль, то на відповідних місцях маски ставлять нулі і використовують команду AND. Якщо необхідно деякі біти даних перевести в одиницю, то в масці використовують одиниці і застосовують команду OR. Команда XOR використовується для інверсії бітів (маска – одиниці) або для знищення (обнуління) бітів (маска – самі дані, наприклад XOR AX, AX).

4. Операції зсуву

Формат команд зсуву: КОП mem/reg, count, де count визначає кількість зсувів. count задається цілим числом або регістром CL. Прапорець CF приймає біт, який «висувається».

SHR mem/reg, count	логічний зсув вправо 	shr dh, 1
SAR mem/reg, count	арифметичний зсув вправо 	sar [BP+8], cl
SHL mem/reg, count SAL mem/reg, count	логічний/арифметичний зсув вліво 	shl al, 1 sal ax, cl
ROR mem/reg, count	циклічний зсув (ротація) вправо 	ror bx, 1
ROL mem/reg, count	циклічний зсув (ротація) вліво 	rol dl, cl
RCR mem/reg, count	циклічний зсув (ротація) через перенос вправо 	rcr [si], cl
RCL mem/reg, count	циклічний зсув (ротація) через перенос вліво 	rcl bl, 1

5. Команди керування

До команд керування відносяться команди безумовних переходів, команди умовних переходів, команди виклику підпрограм, команди повернення із підпрограм, команди керування циклами, команди переривань, команди керування процесором.

JMP adr/lab	Перехід на іншу адресу/мітку	jmp bx jmp lab1
JC adr	Перехід на іншу адресу, якщо CF=1	jc next
JZ adr	Перехід на іншу адресу, якщо ZF=1	jz lab5
JCXZ adr	Перехід на іншу адресу, якщо CX=0	jcxz lab3
CALL adr	Викликати підпрограму за адресою adr	call input
RET	Повернутися із підпрограми	ret
LOOP adr	Якщо CX>0, то перейти на адресу/мітку Замінює дві команди (dec cx та jnz adr)	loop start
INT number	Викликає підпрограму обробки переривання, на адресу якої вказує вектор number	int 21h
IRET	Повернутися із підпрограми обробки переривання	iret
CLC, CMC, STC	Скинути, інвертувати, встановити CF	clc, cmc, stc
CLD, STD	Скинути, встановити DF	cld, std
CLI, STI	Скинути, встановити IF	cli, sti
NOP	Порожня операція	nop
HLT	Зупинка мікропроцесора	hlt

В таблицях наведено тільки найбільш поширені команди процесора I8086. Повний список команд процесора значно більший. Так, наприклад, тільки команд умовних переходів є близько двох десятків.

Порядок виконання роботи

Частина 1

1. Ознайомитися із способами адресації операндів процесора I8086.
2. Проаналізувати програму data4.asm, знайти у ній різні способи адресації операндів та прийоми роботи з табличними даними, виписати приклади команд із різними способами адресації.

Частина 2

3. Ознайомитися з групами команд процесора I8086.
4. Проаналізувати програму data4.asm, знайти та виписати приклади команд передачі даних, арифметичних команд, логічних команд, команд зсуву, та команд керування.

Додаток

Текст програми для лабораторної роботи № 4-5.

```
assume CS:code, DS:data

code segment
begin:

  mov ax, data            ;data segment
  mov ds, ax              ;pointer setting

  mov ax,0003h            ;set screen mode
  int 10h                 ;clear screen

  mov ah, 09h
  lea dx, InpText         ;adress of InpText -> dx
  int 21h                 ;output ASCII string

call Input                ;first digit -> bl, second digit -> cl

  mov ax, 10              ;row length of table
  mul bl                  ;vertical displacement
  mov bx, ax
  mov si,cx               ;horizontal displacement

  mov al, [offset DecTab+bx+si];element of table

  mov cx, 4               ;shift count
  mov ah,al               ;AL copy
  shr ah, cl              ;4 shifts right
  or ah, 30h              ;AH + 30h
  and al, 0fh             ;only least digit
  add al, '0'             ;AL + 30h

  xchg ah, al             ;digits order change
  mov result, ax          ;result -> OutText

  mov ah, 09h
  lea dx, OutText         ;adress of OutText -> dx
  int 21h                 ;output ASCII string

  jmp Finish

Input proc                ;input decimal number

  mov dx, offset InpBuf   ;begin of buffer
  mov ah, 10              ;input ASCII string
  int 21h

  mov si, offset InpStr   ;ASCII string pointer
  xor bx, bx              ;BX=0
  xor cx, cx              ;CX=0
  mov bl, [si]            ;first byte
  mov digits, bl          ;           `-> OutText
  mov cl, [si+2]          ;third byte
  mov digits+2, cl        ;           `-> OutText
  sub bl, 30h             ;bl-30h - first digit
  and cl, 0fh             ;cl-30h - second digit
  ret
Input endp

Finish:
  mov ax,4C00h            ;finishing
  int 21h                 ;of program

code ends


data segment

InpBuf  db 16,0
InpStr  db '0123456789ABCDEF'   ;Input ASCII string
InpText db 'Input your (N+32) number as 3*2: $'
```

```
OutText db 13,10
digits db '0*0 = '
result dw '00'
OutEnd db '$'

DecTab:
row0 db 10 dup (0)
row1 db 0h,01h,02h,03h,04h,05h,06h,07h,08h,09h
row2 db 0h,02h,04h,06h,08h,10h,12h,14h,16h,18h
row3 db 0h,03h,06h,09h,12h,15h,18h,21h,24h,27h
row4 db 0h,04h,08h,12h,16h,20h,24h,28h,32h,36h
row5 db 0h,05h,10h,15h,20h,25h,30h,35h,40h,45h
row6 db 0h,06h,12h,18h,24h,30h,36h,42h,48h,54h
row7 db 0h,07h,14h,21h,28h,35h,42h,49h,56h,63h
row8 db 0h,08h,16h,24h,32h,40h,48h,56h,64h,72h
row9 db 0h,09h,18h,27h,36h,45h,54h,63h,72h,81h

data ends

end begin
```