Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 4

З дисципліни «Архітектура комп’ютерів-1»

На тему «ОБРОБКА ІНФОРМАЦІЇ В ЕОМ НА ПРОГРАМНОМУ І МІКРОПРОГРАМНОМУ РІВНЯХ»

ВИКОНАВ:

студент 2 курсу ФІОТ

групи ІО-41

Логвинчук А. І.

Залікова - 4120

Варіант – 0011000

ПЕРЕВІРИВ:

доц. Ткаченко В. В

Київ – 2016

**Завдання:**

Вивчити етапи формування системи команд процесорів, різновиди форматів команд та способів адресації операндів. Навчитися розробляти мікроалгоритми і мікропрограми виконання етапів команд з використанням мнемонічного мікроасемблера

Функція: *F = (X+Y)\*Z* (1-ий спосіб множення).

Пряма адресація.

Табл. 3.1. Система команд (для реалізації обчислень потрібні такі команди)

|  |  |  |
| --- | --- | --- |
| Команда | Код операції | Операція |
| in aс, [mem] | 0 0 0 1 | Зчитування з ЗП |
| out [mem],ax | 0 0 1 0 | Запис у ЗП |
| mov [mem],ax | 0 0 1 1 | Пересилання з пам’яті до акумулятора |
| mov aс, [mem] | 0 1 0 0 | Пересилання з акмулятора до пам’яті |
| swap ax,reg | 0 1 0 1 | Обмін регістрів |
| exit | 0 1 1 0 | Вихід |
| or [mem], aс | 0 1 1 1 | Діз’юнкція |
| test aс, [mem] | 1 0 0 0 | Перевірка готовності ЗП |
| jz [mem] | 1 0 0 1 | Умовний перехід |
| mul [mem], aс | 1 0 1 0 | Множення |

**Виконання завдання:**

\ Setting-up

LINK l1:ct

LINK l2:rdm

LINK l3:rdd

LINK ewh:16

LINK M:z,z,z,z,z,z,z,14,13,12,11,z

ACCEPT dev[1]:I,072h,074h,12,2

ACCEPT dev[2]:O,0F2h,0F4h,12,2

ACCEPT dev\_buf[1]:0002h,0000000000000101%,0000000000001111%

ACCEPT R7:0020h

ACCEPT R10:03ffh

\ Program in memory

DW 0020h:0001110001010110% \test [0056h]

DW 0022h:0001010001011010% \jz [005Ah]

DW 0024h:0010110001010110% \in r15,[0056h]

DW 0026h:0011110001100100% \mov [0064h],r15

DW 0028h:0001110001010110% \test [0056h]

DW 002Ah:0001010001011100% \jz [005Ch]

DW 002Ch:0010110001010110% \in r15,[0056h]

DW 002Eh:0011110001100110% \mov [0066h],r15

DW 0030h:0001110001010110% \test [0056h]

DW 0032h:0001010001011110% \jz [005Eh]

DW 0034h:0010110001010110% \in r15,[0056h]

DW 0036h:0011110001110110% \mov [0068h],r15

DW 0038h:0100010001100110% \or r15,[0066h]

DW 003Ah:0100110001100100% \mul [0064h]

DW 003Ch:0001110001011000% \test [0058h]

DW 003Eh:0001010001100000% \jz [0060h]

DW 0040h:0010010001011000% \out [0058h]

DW 0042h:0000110000000000% \swap r15,r13

DW 0044h:0001110001011000% \test [0058h]

DW 0046h:0001010001100010% \jz [0062h]

DW 0048h:0010010001011000% \out [0058h]

DW 004Ah:0101010000000000% \exit

\ Data in memory

DW 0056h:006Ah \(IN)

DW 0058h:006Ch \(OUT)

DW 005Ah:0020h \ Return addr

DW 005Ch:0028h

DW 005Eh:0030h

DW 0060h:003Eh

DW 0062h:0046h

DW 0064h:000Fh \(X)

DW 0066h:0005h \(Y)

DW 0068h:0002h \(Z)

DW 006Ah:0072h \IN

DW 006Ch:00F2h \OUT

ORG 0h \ Start addr

{cjp nz,go;}

ORG 2h \ swap

{cjp nz,swap;}

ORG 4h \ addr jz [mem]

{cjp nz,zjump;}

ORG 6h \ addr test [mem]

{cjp nz,testdevice;}

ORG 8h \ addr out [mem],r15

{cjp nz,output;}

ORG 0Ah \ addr in r15,[mem]

{cjp nz,input;}

ORG 0Ch \ addr mov r15,[mem]

{cjp nz,movToReg;}

ORG 0Eh \ addr mov [mem],r15

{cjp nz,movToMem;}

ORG 10h \ addr orr r15,[mem]

{cjp nz,orr;}

ORG 12h \ addr mul

{cjp nz,mul;}

ORG 14h \ addr exit

{cjp nz,exit;}

ORG 20h

go

\ read command from memory

{or nil,r7,r7;ewl;oey;}

{xor nil,r7,r7;ewh;oey;}

{r;cjp rdm,cp;or r8,z,bus\_d;}

\ analyze command format

{and nil,r8,8000h;load rm,flags;}

{cjp not rm\_z,exit;}

\ analyze address mode

{and nil,r8,400h;load rm,flags;}

{cjp rm\_z,exit;}

\ separete operand address

{and r14,r8,03ffh;}

\ jump to opcode

{or nil,r8,z;oey;JMAP;}

\ load subroutine from memory

FromMem

{or nil,r14,z;ewl;oey;}

{r;or r14,bus\_d,z;cjp rdm,cp;}

{or nil,r14,z;ewl;oey;}

{r;or r14,bus\_d,z;cjp rdm,cp;}

{crtn nz;}

\ swap r15,r13

swap

{or r11,r15,z;}

{or r15,r13,z;}

{or r13,r11,z;}

{cjp nz,next;}

\ multiplying

mul

{cjs nz,FromMem;}

{or r11,z,17;}

{xor nil,r14,r15; load rm,flags;}

{or nil,r14,r14; cjp not no ,ll1;}

{sub r14, 0,r14,nz;}

ll1 {or nil,r15,r15; cjp not no ,ll2;}

{xor r15,r15,8000h;}

ll2 {cjp not rm\_c,ll3;}

{add r13,r13,r14,z;}

ll3 {or srl,r13,r13,z;}

{or sr.9,r15,r15,z;}

{sub r11,r11,z,z; load rm, flags;cem\_c;cem\_n;}

{cjp not rm\_z, ll2;}

{or r0, r0, 00f2h;}

{or nil, r0, z; oey; ewl;}

ll4{cjp rdm, ll4; w; or nil, r15, z; oey;}

{cjp not rm\_n,fin;}

{xor r15,r15,0FFFFh;}

{xor r13,r13,0FFFFh;}

{add r15,r15,1,z;}

{add r13,r13,0,rm\_c;}

fin {cjp nz,next;}

\ Command mov r15,[mem]

movToReg

{cjs nz,FromMem;}

{or r15,r14,z;}

{cjp nz,next;}

\ Command mov [mem],15

movToMem

{or nil,r14,z;ewl;oey;}

{r;or r14,bus\_d,z;cjp rdm,cp;}

{or nil,r14,z;ewl;oey;}

{w;cjp rdm,cp;or nil,r15,z;oey;}

{cjp nz,next;}

\ Command or r15,[mem]

orr

{cjs nz,FromMem;}

{add r15,r15,r14;}

{cjp nz,next;}

\ Command jz [mem]

zjump

{cjp not rn\_z,no\_j;}

{cjs nz,FromMem;}

{sub r7,r14,1;}

no\_j {cjp nz,next;}

\ Command in r15,[mem]

input

{cjs nz,FromMem;}

{add r14,r14,2,z;}

{or nil,r14,z;ewl;oey;}

{i;cjp rdd,cp;or r15,bus\_d,z;}

{cjp nz,next;}

\ Command out [mem],r15

output

{cjs nz,FromMem;}

{add r14,r14,2,z;}

{or nil,r14,z;ewl;oey;}

{o;cjp rdd,cp;or nil,r15,z;oey;}

{cjp nz,next;}

\ Command test [mem]

testdevice

{cjs nz,FromMem;}

{or nil,r14,z;oey;ewl;}

{i;cjp rdd,cp;or r14,bus\_d,z;}

{or nil,r14,z;load rn,flags;}

{cjp nz,next;}

\ Load next command address

Next

{add r7,r7,2,z;}

{cjp nz,go;}

\ END

exit

**Висновок**:

Системою команд процесора називають сукупність всіх команд , які може виконувати процесор.

*Команда* – інформаційне слово, що в загальному випадку містить такі дані:

- операція, що виконується з операндами;

- місце розташування операндів в пам’ятті;

- місце запису результату операції;

- спосіб визначення адреси наступної команди.