Practica 5

Objectius

Els objectius d'aquesta practica son conèixer els modes d'adreçament del 8085 i familiaritzar-se amb les seves instruccions.

Explicació de la practica

En aquesta practica fem diversos exercicis amb el simulador i8085 i responem a una sèrie de preguntes, primer fem dos programes molt similars que sumen 2 matrius, i després agafem aquest programa i codifiquem el resultat mitjançant una subrutina.

Informe:

Exercici 1

Pregunta 1:

En què simplificaria molt el codi del programa un dels modes d'adreçament del simulador Ripes?

El codi en Ripes es simplificarà ja que no s'utilitza l'acumulador, el que faríem seria sumar 2 registres amb els valors de mat1 i mat2, i guardar-los en un altre registre.

Calculeu les mides del codi del vostre programa i el nombre de cicles per a la seva execució.

1: Codi guardant el resultat en mat2:

```
.define
      contador 5

.data 50h
      mat1: db 1,2,3,4,5
      mat2: db 6,7,8,9,0
      mat3: db 0,0,0,0,0

.org 100h
lxi h, mat1
lxi b, mat2
mvi d, contador
loop:
      mov e, M
      ldax b
      add e
      stax b
      inx h
      inx b
      dcr d
      jnz loop
hlt
```

Fora del loop tenim 27 cicles i dins del loop, si es salta 51 i si no salta 48. Tenim 5 iteracions, en 4 salta i en 1 no. També s'han de sumar els cicles del hlt, per tant 27 + 4 * 51 + 48 + 4 = 283 cicles.

Les tres primeres instruccions ocupen 8 bytes, i dins del loop 10, més 1 del hlt. En total 19 bytes.

2: Codi guardant el resultat en mat3

```
.define
      contador 5

.data 50h
      mat1: db 1,2,3,4,5
      mat2: db 6,7,8,9,0
      mat3: db 0,0,0,0,0

.org 100h
lxi h, mat1
lxi b, mat3
mvi d, contador

movement:
      mov a, M
      stax b
      inx h
      inx b
      dcr d
jnz movement
mvi d,contador
lxi b,mat3

loop:
      mov e, M
      ldax b
      add e
      stax b
      inx h
      inx b
      dcr d
      jnz loop
hlt
```

A l'inici tenim 27 cicles, al moviment si fa el salt son 40 cicles i si no fa el salt son 37 cicles, després tenim mvi i lxi que son 17 cicles, al loop si fa el salt son 51 cicles i si no fa el salt son 48 cicles, més el hlt que son 4 cicles. Per tant el numero total de cicles serà 27 + 4 * 40 + 37 + 17 + 4 * 51 + 48 + 4 = 494 cicles.

Les primeres tres instruccions ocupen 3 bytes, dins del movement 8, després 5, al loop 10 més 1 del hlt. En total 27 bytes.

Pregunta 2

Quants cicles de rellotge triga en executar-se una instrucció aritmètic – lògica qualsevol? Feu servir el fitxer adjunt on especifica el ISA del 8085. Indica quina és la mida mitjana de les teves instruccions. Calcula els cicles per instrucció mitjà per aquests codis.

Una instrucció aritmètic – lògica qualsevol triga 4 cicles.

Pel primer codi s'executen 44 instruccions en 283 cicles, per tant els cicles per instrucció es de 6,43 cicles/instrucció. I la mida mitjana de les instruccions es de 0,43 bytes/instrucció, ja que ocupa una memòria de 19 bytes.

Pel primer codi s'executen 76 instruccions en 494 cicles, per tant els cicles per instrucció es de 6,5 cicles/instrucció. I la mida mitjana de les instruccions es de 0,36 bytes/instrucció, ja que ocupa una memòria de 27 bytes.

Pregunta/Tasca 3

Pugeu el vostre codi i marqueu quina és la instrucció del vostre programa que triga més cicles en executar-se

1: Codi guardant el resultat en mat2:

```
.define
      contador 5

.data 50h
      mat1: db 1,2,3,4,5
      mat2: db 6,7,8,9,0
      mat3: db 0,0,0,0,0

.org 100h
lxi h, mat1
lxi b, mat2
mvi d, contador
loop:
      mov e, M
      ldax b
      add e
      stax b
      inx h
      inx b
      dcr d
      jnz loop
hlt
```

lxi que son 10 cicles i

jnz quan es realitza el salt que

també son 10 cicles

2: Codi guardant el resultat en mat3

```
.define
      contador 5

.data 50h
      mat1: db 1,2,3,4,5
      mat2: db 6,7,8,9,0
      mat3: db 0,0,0,0,0

.org 100h
lxi h, mat1
lxi b, mat3
mvi d, contador

movement:
      mov a, M
      stax b
      inx h
      inx b
      dcr d
jnz movement
mvi d,contador
lxi b,mat3

loop:
      mov e, M
      ldax b
      add e
      stax b
      inx h
      inx b
      dcr d
      jnz loop
hlt
```

Pregunta/Tasca 4

Traduiu el codi per fer-lo servir amb el simulador Ripes. Quants cicles triga en executar-se? Compareu els resultats (mida de codi, accessos a memòria i clicles promig per instrucció) amb els valors obtinguts per l'i8085

1: Codi guardant el resultat en mat2:

```
.data
mat1: .word 1, 2, 3, 4, 5   #Vector mat1
mat2: .word 6, 7, 8, 9, 0   #Vector mat2
mat3: .word 0, 0, 0, 0, 0   #Vector mat3
cont: .word 5               #Contador

.text

main:
    la a0, cont     #Se carga el valor de la entiqueta cont en a0
    lw a4, 0(a0)    #Se carga el contenido al que apunta a0 en a4
    la a0, mat1     #Se carga mat1 en a0
    la a1, mat2     #Se carga mat2 en a1

loop:
    lw a2, 0(a0)    #Se carga el contenido al que apunta a0 en a2
    lw a3, 0(a1)    #Se carga el contenido al que apunta a1 en a3
    add a3, a2, a3  #Se suma el a2 mas a3 i se guarda en a3
    sw a3, 0(a1)    #Se guarda el contenido de a3 en a1
    addi a0, a0, 4  #La direccion de memoria a la que apunta a0 aumenta 4 bytes
    addi a1, a1, 4  #La direccion de memoria a la que apunta a1 aumenta 4 bytes
    addi a4, a4, -1 #El contador decrementa en uno

    bgtz a4, loop   #Si a4 es mayor a 0 se salta a la etiqueta loop
end:
    nop             #Fin del programa
```

El programa triga 65 cicles en executar-se.

Els cicles promig per instrucció son 65/44 = 1,48 cicles/instrucció

Fa 19 accessos a memòria.

Te una mida de 4896 bits.

Valid bits: 32
Dirty bits: 32
LRU bits: 0
Tag bits: 736
Data bits: 4096

Total: 4896 Bits

2: Codi guardant el resultat en mat3

```
.data
mat1: .word 1, 2, 3, 4, 5   #Vector mat1
mat2: .word 6, 7, 8, 9, 0   #Vector mat2
mat3: .word 0, 0, 0, 0, 0   #Vector mat3
cont: .word 5               #Contador

.text

main:
    la a0, cont     #Se carga el valor de la entiqueta cont en a0
    lw a5, 0(a0)    #Se carga el contenido al que apunta a0 en a5
    la a0, mat1     #Se carga mat1 en a0
    la a1, mat2     #Se carga mat2 en a1
    la a2, mat3     #Se carga mat3 en a2

loop:
    lw a3, 0(a0)    #Se carga el contenido al que apunta a0 en a3
    lw a4, 0(a1)    #Se carga el contenido al que apunta a1 en a4
    add a4, a3, a4  #Se suma el a3 mas a4 i se guarda en a3
    sw a4, 0(a2)    #Se guarda el contenido de a4 en a2
    addi a0, a0, 4  #La direccion de memoria a la que apunta a0 aumenta 4 bytes
    addi a1, a1, 4  #La direccion de memoria a la que apunta a1 aumenta 4 bytes
    addi a2, a2, 4  #La direccion de memoria a la que apunta a2 aumenta 4 bytes
    addi a5, a5, -1 #El contador decrementa en uno

    bgtz a5, loop   #Si a4 es mayor a 0 se salta a la etiqueta loop
end:
    nop             #Fin del programa
```

El codi triga 72 cicles en executar-se.

Els cicles promig per instrucció són 72/55 = 1,3 cicles/instrucció.

Fa 20 accessos a memòria.

Té una mesura de 4896 bits.

Valid bits: 32
Dirty bits: 32
LRU bits: 0
Tag bits: 736
Data bits: 4096

Total: 4896 Bits

Exercici 2: Subrutines

Pregunta 4

Quina instrucció fem servir en tots dos casos per assignar la posició inicial al registre SP?

La instrucció que fem servir es el SPHL.

Pregunta 5

Quina es la instrucció utilitzada per guardar el PC en la pila quan treballem amb subrutines? I per recuperar de nou el valor del PC?

Per guardar el PC en la pila s'utilitza el call, i per recuperar-lo fem servir el return.

Tasca 2

```
.define
    contador 5
    clau 10

.data 50h
    mat1: db 1,2,3,4,5
    mat2: db 6,7,8,9,0
    mat3: db 0,0,0,0,0

.org 100h
lxi h, mat1
lxi b, mat3
mvi d, contador
movement:
    mov a, M
    stax b
    inx h
    inx b
    dcr d
jnz movement
mvi d, contador
lxi b, mat3
loop:
    mov e, M
    ldax b
    add e
    stax b
    inx h
    inx b
    dcr d
    jnz loop
    call codifica

acaba:
    hlt

codifica:
    lxi h, mat3
    mvi d, contador
    mvi b, clau

loop2:
    mov a, M
    xra b
     mov M, a
    inx h
    dcr d
    stax d
    jnz loop2
    jmp acaba
```

Teorico-Practic:

Indiqueu quines són les entrades que ha de tenir la U.C. i quines són les sortides per executar aquesta instrucció

Primera instrucció:

auipc rd, immediate

- PC<= PC+4 Enable Write
- X10 <= PC+(Imm <<= 12 bits)
- ALU <= ADD
- M0 <=0
- M1 <=1
- M2 <=0
- M3<=01
- DWR<=A
- DLR1 <=x
- DLR2 <=x

Segona instrucció:

Addi rd, rs1, immediate

- PC<= PC+4 Enable Write
- X10 <= [X10] + Imm
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=0
- M3<=01
- DWR<=A
- DLR1 <=A
- DLR2 <=x

Tercera instrucció:

lw rd, offset(rs1)

- PC<= PC+4 Enable Write
- X11 <= [@([X10] + Imm)]
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=0
- M3<=01
- DWR<=B
- DLR1 <=A
- DLR2 <=x

Cinquena instrucció:
add rd, rs1, rs2

- PC<= PC+4 Enable Write
- X13 <= [X11] + [X12]
- ALU <= ADD
- M0 <=0
- M1 <=0
- M2 <=1
- M3<=01
- DWR<=D
- DLR1 <=B
- DLR2 <=C

Vuitena instrucció:

Sw rs2, offset(rs1)

- PC<= PC+4 Enable Write
- @([X10] + Imm)<= X13 Mem Write enabled
- ALU <= ADD

- M0 <=0
- M1 <=0
- M2 <=0
- M3<=01
- DWR<=0
- DLR1 <=A
- DLR2 <=D

Conclusió

Amb aquesta practica he començat a entendre com funciona el simulador i8085 i les seves instruccions, com funciona l'acumulador, els parells de registres entre altres funcionalitats.

A la part final de la practica he vist el funcionament de les subrutines i quin paper tenen en el programa principal.