Pràctica 6: Pila i entrada/sortida

Introducció

Objectius:

• Estudi de l'espai de memories del microprocessador

• Comprendre el funcionament de la pila

• Veure un exemple d'utilitzacio de subrutines.

Part 1

Qüestions:

1. **L'adreçament de la instrucció LXI és:**

   c) Immediat

2. **Quina instrucció guarda el PC a la pila?**

   c) CALL

Pregunta 1. Quin espai ocupa en memòria la subrutina sumar?

La subrutina suma ocupa l'espai de memòria des de la posició 619h fins a la posició 620h, és a dir, 8 bytes de memòria.

Pregunta 2. Quants cicles tarda en executar-se la subrutina sumar?

Comptabilitzem totes les instruccions que té:

PUSH => 12 cicles
LDAX => 7 cicles
ADD M => 7 cicles
STAX => 7 cicles
INX => 6 cicles
INX => 6 cicles
POP => 10 cicles
RET => 10 cicles

Observem que tenim en compte el RETURN i que només comptabilitzem un sol cop que s'executa (no sumem tots els cops que s'executa). Fan un total de 65 cicles de rellotge.

Tasca 1

Dibuixeu el mapa de memoria de dades: direccions i contingut. Indiqueu les instruccions que modifiquen les dades de la memoria. En cadascuna d'elles, indiqueu quines modificacions es produeixen. Situeu la memoria de programa. Dins d'ella, localitza el sub-bloc que pertany a la subrutina 'suma'.

Per fer-ho més fàcil i entenedor, en lloc de dibuixar tota la memòria (gairebé tota està buida) només posaré les parts que tenen codi implementat.

.data 00h

Posició	Contingut
00h	1h
01h	2h
02h	3h
03h	4h
04h	00h - 4h (*)
05h	00h - 6h (*)

Les posicions 00h i 01h => mat1 (1h i 2h)

Les posicions 02h i 03h => mat2 (3h i 4h)

Les posicions de memòria 04h i 05h => mat3 i tenen els valors 00h i 00h al principi, però després són modificats (a la rutina SUMAR mitjançant la instrucció STAX D) i guarden els valors 4h i 6h.

Posició	Contingut
0016h	00h – guarda ACC i STT
0017h	00h – guarda ACC i STT
0018h	00h – guarda el PC
0019h	00h – guarda el PC
0020h	08h
0021h	09h

Les posicions 0028h i 0029h corresponen al pila: db 8, 9 que guardem a memòria els valors 8h i 9h. Tanmateix, amb la instrucció SPHL situem la pila a la posició de memòria 0028h, per tant, quan cridem el CALL SUMA, guarda el PC a les posicions de memòria 0027h i 0026h. Seguidament guardem el valor de l'acumulador i del registre d'estats mitjançant la instrucció PUSH PSW, que es guarden a les posicions 0025h i 0024h.

Instruccions que modifiquen dades a la memòria:

- **STAX D,** la cridem per guardar el resultat de la suma sobre la tercera matriu;
- **CALL SUMA** que guardarà el PC a la memòria;
- **PUSH PSW** que guardarà l'ACC i el registre d'estats a memòria.

Aquestes últimes necessiten el RET i el POP respectivament per recuperar les dades.

Seguidament posarem el cos del programa i identificarem on està el bloc SUMA.

Posició	Contingut
600	LXI H, 0028h
601h	
602h	
603h	SPHL
604h	MVI A, 17h
605h	
606h	SUI 2Ah
607h	
608h	MVI B, 02h
609h	
60Ah	LXI D, 0000h
60Bh	
60Ch	
60Dh	LXI H, 0002h
60Eh	
60Fh	
610h	CALL 0619 (LOOP)
611h	
612h	
613h	DCR B
614h	JNZ 610h
615h	
616h	
617h	NOP
618h	HLT
619h	PUSH PSW (SUMA)
61Ah	LDAX D
61Bh	ADD M
61Ch	STAX D
61Dh	INX H
61Eh	INX D
61Fh	POP PSW
620h	RET

619h a 620h => SUMA.

610h a 614h => LOOP.

La resta de posicions especificades són del cos del programa principal, sense subrutines.

Tasca 2

Recordem que la pilla creix en sentit contrari es a dir que quan augmenta, les posicions decreixen.

INSTRUCCIÓ	DESCRIPCIÓ	CANVI EN LA PILA
PUSH PSW	Afegeix PSW a la pila	+ 2 posicions
CALL SUMA	Guardem el PC	+ 2 posicions
POP	Recuperem els registres	- 2 posicions
RETURN	Recuperem el PC	- 2 posicions

Part 2

Objectius:

• Comprendre l'adrecament a ports E/S.

El simulador permet comunicar el processador amb una serie de ports E/S:

• Ports d'entrada: interruptors i teclat.

• Ports de sortida: panell de LEDs, *display* 7 segments, *display* 15 segments.

Per accedir a aquests recursos, haurem d'accedir a les direccions adients dels ports.

Analitzem el següent programa:

```
.data 100h
        pila:
.org 24h
        JMP ports
.org 500h
        LXI H, pila
        SPHL
        CALL ports
        NOP
        HLT
ports:
        PUSH PSW
        in 04h
        ANI 00000001
        OUT 05h
        POP PSW
        RET
```

Tasca 3

Que fa la subrutina 'ports'? Per aixo, introduiu dades amb els interruptors o amb el teclat; observeu en un port de sortida el resultat de la subrutina.

La rutina "ports" agafa el valor que introduïm pel port 04h, que tenim per defecte el panell d'interruptors i ho posa a l'acumulador. Es fa un AND amb el valor guardat a l'acumulador i el valor 00000001 i es deixa a l'acumulador. Aquest valor és mostrat pel port 05h mitjançant la instrucció OUT 05h. A més, abans havíem guardar el contingut de l'acumulador i el registre d'estats amb la operació PUSH PSW i ara ho recuperem amb el POP.

Part 3

Objectius:

• Dissenyeu un programa *assembler* que representi en un *display* de 7 segments els numeros del 0 al 5. Els numeros s'introduiran a partir del teclat. A mes, ha de permetre l'opcio d'esborrar el *display*; per aixo, en premer la lletra 'c', es produira un CLEAR del *display*.

Propostes de millora:

El simulador 8085 te una part de la memoria de dades que implementa una 'Pantalla de text'. Modifiqueu el programa per escriure la lletra que vulgueu a la pantalla. Al menu "Opciones" → "de interrupciones" podeu habilitar que es generi una interrupcio TRAP en premer qualsevol tecla del teclat. Aixo produira el salt de l'execucio del programa a la subrutina que hi hagi a la direccio de memoria 0024h. Aprofiteu aquesta possibilitat per reproduir el funcionament continu del teclat i la pantalla.

TASCA 4 (cal entregar)

Pujeu un fitxer amb:

• Explicacio breu de l'algorisme triat.

• Programa assemblador comentat.

Primer mirem quina combinació de nombres binaris correspon a cada nombre que volem representar al display de set segments, de manera que:

- **zero:** 01110111b
- **un:** 01000100b
- **dos:** 00111110b
- **tres:** 01101110b
- **quatre:** 01001101b
- **cinc:** 01101011b

Mirem quins valors corresponen als nombres entrats per teclat, des del 0 fins al 5 i després la C: 30h, 31h, 32h, 33h, 34h, 35h i 43h. Com veiem, no es pot trobar una funció que ens expressi directament una correspondència entre els valors que necessitem per representar-los al display de 7 segments i els valors que prenen al ser introduïts per teclat.

Solució: guardar a memòria aquests valors i fer una funció que, donat un valor per teclat (entre 30h i 35h en el cas dels nombres i el valor 43h en el cas de la 'C'), accedeixi a la posició de memòria on està guardat el valor corresponent a ensenyar per el display i el carregui a l'acumulador per poder-lo ensenyar mitjançant la instrucció OUT.

El programa ha de començar a la posició 0024h, que és on es troba el codi que s'executa quan es detecta la interrupció TRAP. Però no només volem introduir un nombre, per tant, el final del codi seria un bucle infinit:

```
Loop:
    JMP loop
```

D'aquesta manera, el programa no acaba mai a no ser que nosaltres així ho indiquem. Així, quan introduim un altre nombre el programa tornara a saltar a la posició 0024h.

El codi és el següent:

```
.data 00h                ;valors necessaris per al display
        zero: db 01110111b
        un: db 01000100b
        dos: db 00111110b
        tres: db 01101110b
        quatre: db 01001101b
        cinc: db 01101011b
.data 13h                ;valor per a netejar el display
        clear: db 00000000b
.org 24h
        IN 00h           ;obtenim el valor del port 00h i el carreguem a
                         ;l'acumulador.
        SUI 30h          ;li restem 30 per obtenir la posició de memòria on hem
                         ;guardat les dades per ensenyar-ho al display
        MOV C, A         ;ACC => BC
        LDAX B           ;Carreguem a l'acumulador els valors corresponents a la
                         ;posició de memòria del contingut dels registres BC
                         ;havíem guardat el valor introduït – 30h per obtenir la
                         ;posició de memòria adequada, estem accedint a un
                         ;valor a ensenyar
        OUT 07h          ;carreguem el valor de l'acumulador al display de 7
                         ;segments.
loop:                    ;bucle infinit
        JMP loop
HLT                      ;acabem
```

Conclusions

En aquesta pràctica hem estudiat la distribució de memòria del microprocessador i el funcionament dels dispositius d'entrada sortida.

- Hem analitzat el mapa de memòria d'un codi.
- Hem analitzat el funcionament dels dispositius entrada/sortida.
- Hem fet un petit programa que ensenya uns valors pel display de 7 segments.