Introducció als ordinadors Pràctica 6 Joan Viñas Simon **PRÀCTICA 6: Pila i entrada/sortida**

Introducció i objectius:

Aquesta pràctica ens servirà per aprofundir en els modes d'adraçament en aquest nou si mulador i8085, i entendre la distribució de memòria del microprocessador. També veurem el funcionament i com podem manipular les piles, i seguirem treballant amb el concepte de subrutines. Finalment, veurem un exercici on usarem dispositius d'entrada com per exemple, un teclat, i de sortida, una pantalla display de 7 segments.

Part 1

Cas on el resultat es guarda en mat3:

```
i8085
.define
 num 02h
.data 00h
 mat1: db 1,2
 mat2: db 3,4
 mat3: db 0,0
.data 20h
 pila:
.org 600h
 LXI H, pila ;carreguem 20h a HL
 SPHL ;la pila comença a @[H, L] = 20h
 MVI B, num ;A <- num = 2
 LXI D, mat1 ;DE <- mat1
 LXI H, mat2 ;HL <- mat 2
 LXI B, mat3 ;BC <- mat3
loop:
 CALL suma ;cridem la instruccio per sumar
 DCR A ;li restem 1 a B, mentre no sigui zero, ens quedem en bucle   JNZ loop
 NOP
 HLT
suma:
 PUSH PSW ;guardem l'ACC i el registre d'estats a la pila
 LDAX D ;A <- [DE]
 ADD M ;A += [HL]
 STAX B ;BC <- A
 INX H ;HL++
 INX D ;DE++
 INX B ;BC++
 POP PSW ;Fem pop de l'acumulador i el registre d'estat.
 RET
```

Aquest programa fa la suma dels dos vectors (mat 1 + mat2), i ho guarda al vector mat3.

Pregunta 1

L'adreçament de la instrucció LXI és **immediat**.

Pregunta 2

Quina instrucció guarda el PC a la Pila?
La instrucció utilitzada per **guardar** el PC a la pila és la CALL.

Pregunta 3
Quin espai ocupa en memòria la subrutina ‘suma’?
PUSH PSW ;1 byte
LDAX D ;1 byte
ADD M ;1 byte
STAX D ;1 byte
INX H ;1 byte
INX D ;1 byte
POP PSW ;1 byte
RET ;1 byte
La subrutina ‘suma’ ocupa un total de 8 bytes.

Pregunta 4
Quants cicles triga en executar-se la subrutina ‘suma’?
PUSH PSW ;12 cicles
LDAX D ;7 cicles
ADD M ;7 cicles
STAX D ;7 cicles
INX H ;6 cicles
INX D ;6 cicles
POP PSW ;10 cicles
RET ;10 cicles
La subrutina ‘suma’ triga un total de 65 cicles.

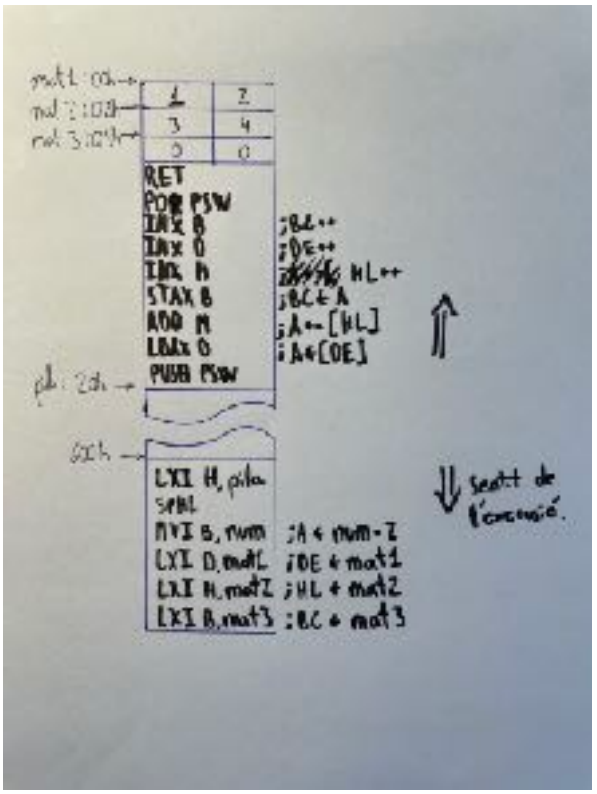

Fig 1. Distribució de les instruccions i dades al microoprocesador

POSICIÓ	CONTINGUT
000h	mat1: 1
001h	2
002h	mat2: 3
003h	4
004h	mat3: 0
005h	0
020h	pila:
600h	LXI H, 0020h
601h	
602h	
603h	SPHL
604h	LXI D, mat1
605h	
606h	
607h	LXI H, mat2
608h	
609h	
60Ah	LXI B, mat3
60Bh	
60Ch	
60Dh	CALL SUMA
60Eh	

60Fh	
610h	DCR A
611h	JNZ loop
612h	NOP
613h	HLT

614h	PUSH PSW
615h	LDAX D
616h	ADD M
617h	STAX B
618h	INX H
619h	INX D
61Ah	INX B
61Bh	POP PSW
61Ch	RET
61Dh	LXI D, mat1

Fig 2. Mapa de memòria

TASCA 2

El començament de la pila a la memòria es troba a la posició 20h.

Instrucció	**Descripció**	**Canvi en la pila**
PUSH PSW	Afegeix PSW a la pila	Augmenta en 2 bytes.
POP PSW	Recuperar PSW de la pila	Es buida la pila
RET	Recupera l'adreça de retorn.	Disminueix en 2 bytes.
SPHL	Carrega el contingut dels	

	registres HL a la pila.	

Part 2

```
.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
Què fa la subrutina 'ports'?

La rutina "ports" agafa el valor que introduïm pel port 04h, que tenim per defecte al 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 guardat el contingut de l'acumulador i el registre d'estats amb la operació PUSH PSW i ara ho recuperem amb el POP.

Introducció als ordinadors Pràctica 6 Joan Viñas Simon

Part 3

Dissenyeu un programa assembler que representi en un display de 7 segments els núme ros del 0 al 5. Els números s'introduiran a partir del teclat. A més, ha de permetre l'opció d'esborrar el display; per això, en prémer la lletra 'c', es produirà un CLEAR del display.

```
.data 00h
        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     ;posem al parell de registres B, C el valor que teníem a l'acumulador. LDAX B
;Carreguem a l'acumulador els valors corresponents a la posició de         ;memòria del contingut
dels registres B, C
                                    ;havíem guardat el valor introduït – 30h per obtenir la posició de
                                     ;memòria adequada, estem accedint a un valor per ensenyar-lo.
        OUT 07h     ;carreguem el valor de l'acumulador al display de 7 segments. loop:
;loop infinit
        JMP loop
        HLT ;acabem
```

-> Explicació de l'algoris-me triat:

Per començar, s'ha d'analitzar quina combinació binaria correspon a cada nombre deci mal que volem mostrar al display:

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

També ens hem fixat en 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 que no hi ha cap funció que directementens mostri pel display certs valor introduïts per teclat, el que fem és crear una funció, que donat un valor per teclat entre 30h i 35h, i el 43h, accedeixi a la posició de memòria on està guardat el valor corresponent a ensenyar pel display, i que aquest estigui carregat al acc per poder-lo mostrar al display mitjançant la instrucció OUT.

Finalment, creem un bucle infinit amb l'objectiu que el programa mai es pari, i en cas que ho vulguem, el pararem manualment.

Conclusió:

Aquesta pràctica ens ha servit per aprofundir amb el tema de les subrutines, i amb les instruccions que ens permeten manipular les piles, i he vist el que ocupa una pila. També he pogut veure un diagrama d'un mapa de memòria on m'ha ajudat entendre la distribució de memòria del microprocessador. Finalment, he pogut crear un programa on l'output es mostri visualment per pantalla, gràcies a la interacció amb el display.