**Universitat Autònoma de Barcelona** **Escola d'Enginyeria** **23/01/2018** **Fonaments d'Informàtica**

| **Cognoms**: | **NIU**: |
|---|---|
| **Nom:** | **Grup**: |

*Teoria 1.* ***( 1 punt)***

Descriviu breument la funcionalitat dels següents registres:

a) Comptador de programa (PC)

Registre que conté l'adreça de la instrucció en curs d'execució o de la següent, depenent d'implementacions.

b) Registre de transferència de dades a Memòria (MDR) i de transferència d'adreces (MAR)

MDR Registre de dades o Memory Buffer Register conté la dada que es vol escriure a la memòria principal o la que s'ha llegit de la memòria principal.

MAR Memory Address Register conté l'adreça de la memòria principal on es vol fer la lectura o l'escriptura d'una dada.

Ambdós registres estan connectats directament amb els bus, dades i adreces respectivament

c) Registre d'estat (SR)

SR Registre que agrupa bits de caràcter lògic que informen de l'estat de diferents parts de la CPU en particular de la ALU. Bits típics són els de Carry, overflow, signe, etc.

d) Apuntador a la pila/Stack pointer (SP)

L'apuntador de la pila és un registre que conté l'adreça de la capçalera de la pila i que el CPU utilitza per simular, a la memòria principal, l'existència d'una memòria pila que físicament no existeix però que és molt útil per a la gestió de les subrutines. Segons la implementació pot estar apuntant la primera posició a escriure o la darrera on es va escriure. Per la mateixa raó la pila pot créixer cap a les adreces més altes o cap a les més baixes.

*Teoria 2.* ***( 1,5 punts)***

a. Ordeneu els diferents nivells de memòria que podem trobar en un computador d'acord amb la seva capacitat i velocitat.

Memòria secundària, d'una capacitat podríem dir il·limitada (Gigues, Teres) però amb un temps d'accés molt alt (milisegons). Permet emmagatzemar de manera estàtica gran quantitat d'informació destinada a ser mantinguda durant un període indefinit de temps.

Memòria principal, capacitat molt més reduïda (Megues, Gigues) però amb un temps d'accés millor (micro o nanosegons). Destinada a emmagatzemar el programa en execució i les dades i estructures que necessita per a executar-se.

Memòria Cache, capacitat reduïda (kilobytes) i temps d'accés de pocs nanosegons. Destinada a emmagatzemar sèries d'instruccions i segments de dades per a que la CPU les tingui més a l'abast sense necessitat d'anar a buscar-les a la Memòria Principal.

Registres interns de la CPU. Es compten per unitats o desenes, i un accés d'alguns nanosegons. Destinats a mantenir informació molt concreta i específica procedent d'instruccions i dades generades pel propi processador.

El cost per unitat d'emmagatzematge naturalment creix ràpidament des de la mem. Secundària als registres de manera molt accentuada.

a. Descriviu breument la funció de la memòria Cache?

Memòria Cache, És una memòria intermèdia entre la principal (MP) i el processador. Normalment construïda amb la tecnologia mes avançada del moment, degut a l'alt cost ha de ser de mida molt reduïda. Destinada a emmagatzemar sèries d'instruccions i segments de dades per a que la CPU les tingui més a l'abast sense necessitat d'anar a buscar-les a la Memòria Principal.
Durant l'execució del programa en curs, les instruccions i les dades procedents de la MP i que el processador va utilitzant, addicionalment són emmagatzemades a la Cache. Posteriorment, en accedir a dades/instruccions veïnes a les anteriors o quan es tornen a executar (bucles), com que la CPU les busca simultàniament a la Cache i a la MP, les trobarà abans a la cache i l'execució del programa serà més ràpida que contra la MP. Quan la Cache queda plena d'informació hem de recorre a algoritmes de reemplaçament que decideixin quin informació antiga ha de ser substituïda per la nova procedent de la MP.

*Teoria 3.* ***(3 punts)***

Descriviu l'execució de les següents instruccions a nivell de transferència de registres i seguint tots els passos del model de màquina de Von Neumann. (MAR i MDR són els registres de transferència amb els busos de memòria)

a. **add AX, (1000)** (El modes d'adreçament utilitzats són 'de registre directe' i 'indirecte' respectivament)

(Cerca de la instrucció)
MAR ← <PC>
Ordre de lectura + temps d'espera de la lectura
Càrrega del MBR — L'MBR captura la instrucció
IR ← <MBR> instrucció al registre de instruccions
Descodificació + PC ← (PC) +1
MAR ← (Reg Inst) $_{\text{adreça}}$ — EL 1000 passa del RI al MAR
Ordre de lectura + temps d'espera de la lectura
Càrrega del MBR — L'MBR captura el contingut de la posició 1000
MAR ← (MBR) — Iniciem la lectura de l'operand
Ordre de lectura + temps d'espera de la lectura
Càrrega del MBR — L'MBR captura el contingut de la posició (1000)
AX ← (MBR) + <AX> — Execució de la instrucció al Registre AX.

b. **pusx AX** (La pila creix a posicions menors de memòria i l'SP apunta a la primera posició lliure de la pila)

Cerca de la instrucció
Descodificació + PC ← (PC) +1
MAR ← (SP) — EL Stack Pointer passa al MAR
MBR ← (AX) — Pot ser simultània a l'anterior
Ordre d'escriptura + temps d'espera de l'escriptura + SP ← (SP) – 1 Fem créixer la pila avall

c. **mov 1000[SI], BX** (El modes d'adreçament utilitzats són 'indexat' i 'de registre directe' respectivament)

Cerca de la instrucció
Descodificació + PC ← (PC) +1
MAR ← (Reg Inst) adreça + (SI) — Sumem a 1000 el del registre índex
MBR ← (BX) — Pot ser simultània a l'anterior
Ordre d'escriptura + temps d'espera de l'escriptura

*Teoria 4.* ***(1 punt)***

Descriviu els diferents tipus d'instruccions de ruptura de seqüència o salt. Podeu incloure exemples d'ús.

Salt incondicional, "JMP address" Substitueix el contingut del PC per l'adreça efectiva.

Salts condicionals, en funció de si una determinada condició relacionada amb els flags de l'ALU es comporta com l'anterior o segueix en seqüència.

Salt a subrutina, es comporta com un salt incondicional però amb l'afegit de salvar a la pila el PC incrementat en 1 per garantir el posterior retorn al punt d'execució.

Retorn de subrutina, Substitueix el contingut del PC per l'adreça emmagatzemada al TOP de la Pila. Això fa efectiu el retorn a la seqüència principal d'execució.

Retorn d'interrupció, Igual que l'anterior però amb l'afegit de restaurar la paraula d'estat saldada al següent TOP de la Pila i retorn al mode usuari del processador.

*Teoria 5.* ***(1 punt)***

En un moment determinat el processador es troba executant la instrucció cmp R0, R1 que pertany a un cert programa P1. Coneixem les següents dades:

- Aquesta instrucció és a la posició 1000 de memòria,
- R0 i R1 són registres de propòsit general de 8 bits. En el moment d'executar-la R0 conté el valor 10110010 i R1 el valor 11100111 (Expressats en C2)
- A l'adreça 1001 tenim la instrucció sub R1, R0 (sub = resta)
- A l'adreça 1002 hi tenim la instrucció jon 2000 (jon = salta si negatiu)
- El registre d'estat té també 8 bits, però només s'hi emmagatzemen els bits C, Z, S i O (en aquest ordre i en els bits menys significatius del registre), els altres 4 bits sempre valen 0.

Responeu les preguntes següents:

i) Quin és el contingut del registre d'estat després d'executar la instrucció de la posició 1000? (Justifiqueu la vostra resposta)

R0=10110010 R1=11100111
R0-R1= 10110010 + 00011001 = 11001011 C=0, Z=0, S=1, O=0 SR=00000010

ii) Quin és el contingut del registre d'estat després d'executar la instrucció de la posició 1001? (Justifiqueu la vostre resposta)

R0=10110010 R1=11100111
R1-R0= 11100111 + 01001110 = 00110101 C=1, Z=0, S=0, O=0 SR=00001000

iii) Quin és el contingut del registre comptador de programa (PC) després d'executar la instrucció de la posició 1002? (Justifiqueu la vostre resposta).

1003 perquè el resultat darrer no ha estat negatiu

*Teoria 6.* ***( 1,5 punts)***

a) Expliqueu què significa que l'espai d'entrada/sortida estigui "mapejat" a la memòria principal.

Significa que els registres dels controladors dels perifèrics es veuen en adreces que podrien correspondre a la memòria, és a dir que els controlador i la memòria són connectats al mateix bus i per accedir al i/o ports es fa mitjançant instruccions de tips MOVE com a les posicions de memòria.

Dibuix CPU, MP, I/O amb bus únic

b) Descriviu els tres mecanismes d'E/S (E/S programada amb llaços d'espera, E/S programada amb interrupcions i Accés Directe a Memòria). Indiqueu les raons de l'existència dels dos darrers mecanismes.

L'E/S programada de dades (TPD) amb llaços d'espera es fa mitjançant un programa executat per la CPU on es

1 selecciona el dispositiu

2 es determina si esta disponible

3 s'accedeix a l'i/o port

4 si no hem acabat tornar a 2

El pas "2" es fa amb un bucle on es llegeix sistemàticament esperant que estigui disponible. En aquest bucle podem consumir el 99% dels temps de la TPD.

Si canviem aquest bucle pel fet que el controlador enviï una interrupció a la CPU quan estigui disponible serà una E/S programada amb interrupcions. Això comporta que a partir del punt 3 la CPU pot anar a executar altres programes i anirà fent el pas 3 cada cop que rebi una interrupció. El pas 3 serà la rutina de tractament de la interrupció.

Accés Directe a Memòria consisteix en incloure un controlador de DMA. Aquest dispositiu pot substituir completament la CPU en fer la transferència mentre la CPU fa altres tasques. El controlador de DMA quan acabi la feina que se li ha encomanat d'entrada o sortida, interromprà la CPU per a que li programi una altra tasca. Per tant la intervenció de la CPU es limita a programar el que ha de fer el DMA i el DMA fa la transferencia.

*Teoria 7.* ***(1 punt)***

Responeu raonadament les preguntes següents.

a) Què són els modes usuari i supervisor?

És una característica hardware dels processadors moderns que determina quins recursos té disponibles el programa que s'estigui executant.

El mode usuari restringeix el conjunt d'instruccions que pot executar el programa, el conjunt d'adreces de memòria a que pot accedir i l'impedeix fer E/S.

En el mode supervisor el programa en execució té accés a tots els recursos del computador

b) En quin moment es produeix el canvi entre mode usuari i mode supervisor i a l'inrevés?

En el moment que es produeix una interrupció i es decideix atendre-la, un cop desat l'estat del programa que en execució, es fa el canvi de mode usuari a supervisor.

En el moment que el SO torna el control a un altre programa mitjançant la instrucció "retorn d'interrupció" es passa de mode supervisor a mode usuari

c) Quin és l'únic software que té accés a tots els recursos del computador? Quina és la conseqüència principal d'aquest fet des del punt de vista de la gestió i us d'aquests recursos?

El SO perquè un dels seus objectius principals és el de gestionar els recursos del computador. Si un programa d'usuari pogués executar-se en mode supervisor aleshores podria gestionar recursos sense la participació del SO i impediria que aquest complís el seu objectiu.

d) Quin és el nom del mecanisme que permet que un programa usuari demani fer una operació d'E/S?

Com el programa usuari s'executa en mode usuari i no pot accedir als dispositius d'E/S, ha de disposar d'un mecanisme que li permeti demanar l'operació a l'únic software que hi té accés (el SO). Com el SO només rep el control del sistema quan es produeix una interrupció, el programa usuari ha de provocar una interrupció voluntària en forma de trap.

Aquest mecanisme es coneix com a crida al sistema

*Teoria 8.* ***(1 punt)***

Des del punt de vista del Sistema Operatiu:

a) Què és un procés?

Un procés és l'activitat desencadenada Per l'execució d'un programa conjuntament amb els recursos que utilitza.

La materialització d'un procés consisteix en el codi del programa, l'estat del processador i els seus registres i piles així com el dels recursos utilitzats (inclosa la memòria) i la informació continguda al Bloc de Control de Procés.

b) Quina estructura de dades es fa servir per gestionar un procés?

El Bloc de Control de Procés (PCB)

c) Quin tipus d'informació s'emmagatzema en aquesta estructura?

Informació d'identificació

- PID del procés, PID del pare
- ID d'usuari real (uid real)
- ID de grup real (gid real)
- ID d'usuari efectiu (uid efectiu)
- ID de grup efectiu (gid efectiu)

Estat del processador

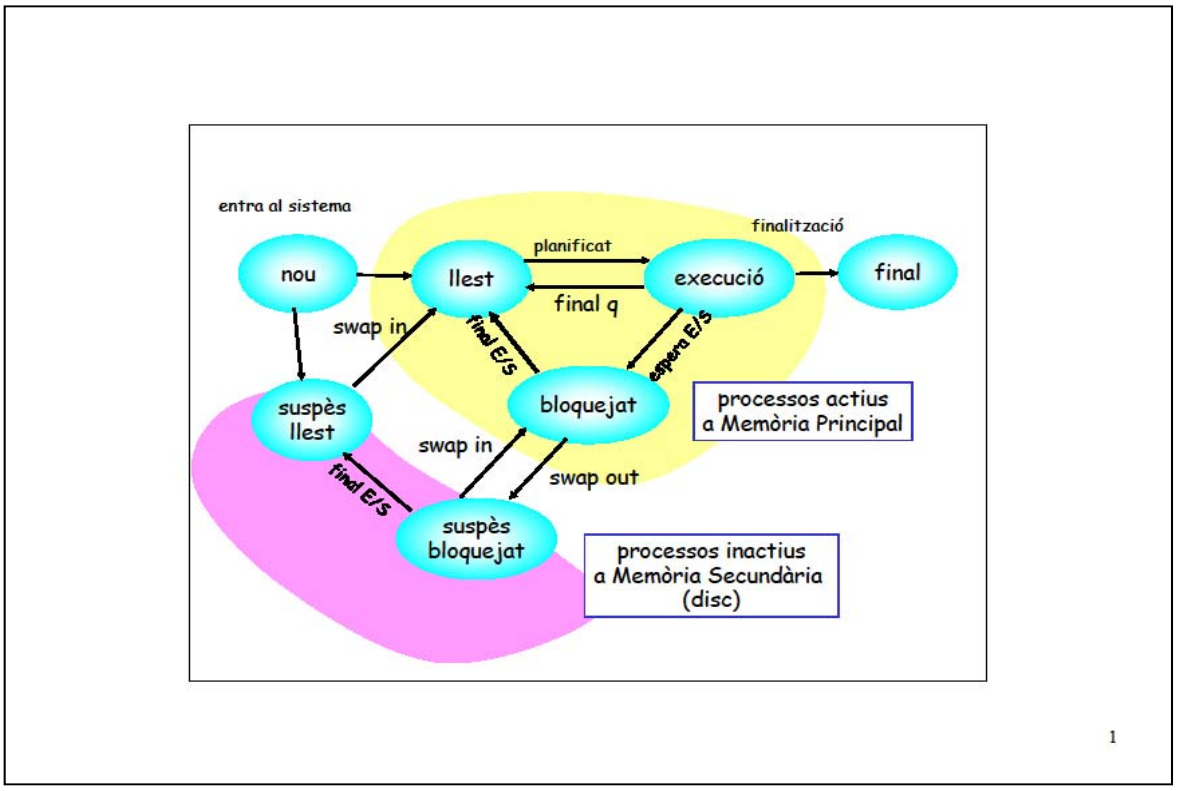
Informació de control del procés

- Informació de planificació i estat
- Descripció dels segments de memòria del procés
- Recursos assignats (fitxers oberts, ...)

Comunicació entre processos.

Apuntadors per estructurar els processos en llistes o cues.

*Teoria 9.* ***(1 punt)*** Dibuixeu el diagrama d'estats complet d'un procés indicant i explicant les transicions possibles entre estats.




Explicar com a mínim els tres estats actius: Llest, execució i bloquejat. I les transcicions

*Problema 1. (2* ***punts)***

A partir del següent fragment de codi en llenguatge màquina, escriviu el fragment de codi equivalent en llenguatge C. En cas que sigui necessari podeu definir noms de variables i el seu tipus més adient.

| **Instruccions** | **Comentari** |
|---|---|
| mov R0, 1000<br>mov R1, 1001<br>mov R2, 1002 | ;Adreçament De registre directe i Directe |
| cmp R0, R1 | ;Comparació |
| jnn <label1> | ;Salta si no negatiu (jge) |
| label 0: cmp R0, R1 | |
| joz <label2> | ;Salta si zero (je) |
| add R2, R0 | ;Suma |
| inc R0 | ;Increment |
| jmp <label0> | ;Salt incondicional+ |
| label1: mul R2, R0 | ;Multiplica |
| label2: add R0, R1 | ;Suma |

```
X (adreça 1000)
Y (adreça 1001)
Z (adreça 1002)
If (X<Y)
        while (X!=Y)
        {
            Z:=Z+X;
            X++;
        }
    else   Z:=Z*X ;
X:=X+Y
```

```
mov 1001, R1
mov 1002, R2
```

*Problema 2. (2 **punts**)*

A partir del fragment de codi següent en llenguatge C, escriviu el fragment de codi equivalent en llenguatge màquina. Cas que sigui necessari pot assignar a cada variable l'adreça que considereu adient.

```
#define N 1000

int i, V[N], A[N],B[N];
...
for ( i = 0; i < N; i++ )
        if ( A[i] <= B[i] ) V[i] = A[i];
        else V[i] = B[i];
```

```
i -> SI     V -> adreça 1000      A-> adreça 2000      B-> adreça 3000
        mov SI,#0
for:    cmp SI,#1000
        jnn    fifor
        mov AX,2000[SI]
        mov BX,3000[SI]
        cmp AX,BX
        jg     else
        mov 1000[SI],AX
        jmp  fiif
else:   mov 1000[SI],BX
fiif:   inc SI
        jmp for
fifor:  ...
```

*Problema 3. (1 **punts**)*

Considerant els següents números expressats en las bases indicades:

Quina és la seva representació en binari fent servir 16 bits i Complement a 2?

**Nota**: Les operacions realitzades formen part de la resposta, si poseu només el resultat tindrà una valoració de 0 punts.

a. $+565_{(10}$

565/2 = 282 R(1)

282/2 = 141 R(0)

141/2 = 70 R(1)

70/2 = 35 R(0)

35/2 = 17 R(1)

17/2 = 8 R(1)

8/2 = 4 R(0)

4/2 = 2 R(0)

2/2 = 1 R(0)

Solució = **$0000\ 0010\ 0011\ 0101_2$** Complement a 2 d'un número positiu és el número positiu.

b. –6BF2(16

-(6 = 0110) (B = 1011) (F = 1111) (2 = 0010)

Conjunt = $-(0110101111110010)_2$

Complement a 2 perquè és negatiu = **$1001\ 0100\ 0000\ 1110_2$**

0000 = 0

0001 = 1

0010 = 2

0011 = 3

0100 = 4

0101 = 5

0110 = 6

0111 = 7

1000 = 8

1001 = 9

1010 = A

1011 = B

1100 = C

1101 = D

1110 = E

1111 = F

*Problema 4. (1* ***punt)***

Considerant els següents números expressats en las bases indicades:

Quina és la seva representació en binari fent servir 16 bits i Complement a 1?
**Nota**: Les operacions realitzades formen part de la resposta, si poseu només el resultat tindrà una valoració de 0 punts.

**a.** $-2132312_{(4}$

$= -(2*4^6 + 1*4^5 + 3*4^4 + 2*4^3 + 3*4^2 + 1*4^1 + 2*4^0)$

$= -(2*4096 + 1*1024 + 3*256 + 2*64 + 3*16 + 1*4 + 2*1)$

$= -(8192 + 10124 + 768 + 128 + 48 + 4 + 2)$

$=-(10166)_{10}$

10166 / 2 = 5083 (0)

5083 / 2 = 2541 (1)

2542 / 2 = 1270 (1)

1270 / 2 = 635 (0)

635 / 2 = 317 (1)

317 / 2 = 158 (1)

158 /2 = 79 (0)

79 / 2 = 39 (1)

39 / 2 = 19 (1)

19 / 2 = 9 (1)

9 / 2 = 4 (1)

4 / 2 = 2 (0)

2 / 2 = 1 (0)

Solució = - $(0010\ 0111\ 1011\ 0110)_2$ Com que és negatiu hem de fer el complement a 1. => **$1101\ 1000\ 0100\ 1001_2$**

**b.** $+7251_{(8}$

000 = 0

001 = 1

010 = 2

011 = 3

100 = 4

101 = 5

110 = 6

111 = 7

111 010 101 001 = Com que són 16 bits ens queda = > **$0000\ 1110\ 1010\ 1001_2$**

*Problema 5. **(2 puns)***

A partit dels següents números de 8 bits, expressats en binari fent servir la representació de Complement a 2:

A: 11110110 i B: 01110110

Calculeu el valor de les expressions operand en binari, sense fer servir l'operació de resta i indicant i justificant si es produeix error de desbordament o no:

a. A + B
b. A – B
c. B – A

a)

```
  11110110  (A)
+ 01110110  (B)
  01101100  (A+B)
```

b)

```
  11110110  (A)
+ 10001010  (-B)
  10000000
```

c)

```
  00001010  (-A)
+ 01110110  (B)
  10000000      OVERFLOW (resultat de diferent signe que el dels 2 operands)
```

**Nota**: Si feu els càlculs correctament val 1 punt, l'altra punt es dona per indicar i justificar correctament si hi ha o no desbordament,

*Problema 6 (2* ***puntos****)*

A partir del següent fragment de programa i l'estat inicial de la memòria, mostreu l'evolució dels continguts dels registres i de la memòria a mesura que s'executi el programa. Suposeu una màquina como la vista a teoria.

| INSTRUCCIÓN | | | | | **Posiciones de memoria** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *PC* | *A* | *X* | *SP* | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | 50 | 51 | 52 |
| Estado inicial | 300 | 5 | 2 | 50 | 5 | 3 | 6 | 4 | 1 | 7 | 7 | *2* | *4* | 9 | 9 | 9 |
| | **301** | **2** | | | | | | | | | | | | | | |
| | **302** | | | | | | | | | **2** | | | | | | |
| | **303** | **7** | | | | | | | | | | | | | | |
| | **304** | | | **51** | | | | | | | | | | **7** | | |
| | **305** | **3** | | | | | | | | | | | | | | |
| | **306** | **6** | | | | | | | | | | | | | | |
| | **307** | | | **52** | | | | | | | | | | | **2** | |
| | **308** | | **7** | | | | | | | | | | | | | |
| | **309** | | **8** | | | | | | | | | | | | | |
| | **310** | | **2** | **51** | | | | | | | | | | | | |
| | **311** | **5** | | | | | | | | | | | | | | |
| | **312** | **4** | | | | | | | | | | | | | | |
| | **313** | **7** | | **50** | | | | | | | | | | | | |
| | **314** | **8** | | | | | | | | | | | | | | |
| | **315** | | | | | | | | | | | | | | | |
| | **316** | | | | | | | | | | | | | | | |
| | **317** | | | | | | | | **8** | | | | | | | |
| | **318** | | | | | | | | | | | | | | | |
| | **400** | | | | | | | | **8** | | | | | | | |
| | **401** | **0** | | | | | | | | | | | | | | |
| | **402** | **5** | | | | | | | | | | | | | | |
| | **403** | **10** | | | | | | | | | | | | | | |
| | **404** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |

```
301     SUB A, 1
302     MOV 5, A
303     MOV A, 4[X]
304     PUSH A
305     SUB A, 8
306     ADD A, #3
307     PUSH X
308     MOV X, (2)
309     INC X
310     POP X
311     SUB A, [X+2]
312     MOV A, (1)
313     POP A
314     ADD A, 2[X]
315     CMP A, 2
316     JOZ 400
317     MOV (3), A
318     JMP 400

400     MOV [X+2], A
401     AND A, #2
402     OR A, #5
403     SHL A
404     END
```

**Nota 1**: Modes d'adreçament utilitzats: # num (immediat), num (directe), num[X] (indexat), (num) (indirecte), [X+num] (Relatiu a base)

**Nota 2**: Instrucció SHL = desplaçament a l'esquerra d'1 bit

**Nota 3**: el registre SP apunta a la primera posició lliure de la pila i aquesta creix cap a posicions majors de memòria.

**Nota 4**: JOZ és equivalent a JE