

PROBLEMES
DE LA
MÀQUINA
RUDIMENTÀRI
A

Fermín Sánchez Carracedo
Anna Maria del Corral González
Enric Pastor Llorens

Problemes de llenguatge màquina

- Traduïu a llenguatge ensamblador de la MR les següents instruccions expressades en hexadecimal.
 - 1006h
 - 4E20h
 - 80A1h
 - EA64h
 - C6F1h
 - E846h
 - A825h
- Passeu a notació binària i hexadecimal les següents instruccions escrites en llenguatge ensamblador de la MR.
 - ADD R0, R1, R2
 - SUBI R4, #1, R5
 - BGE 43
 - BR 12
 - AND R1, R7, R6
 - ASR R4, R2
- Indiqueu quin és el resultat d'executar les instruccions següents, donant el contingut final dels registres i posicions de memòria que es modifiquen i el valor dels indicadors de condició. Se suposa per a cada cas que el contingut inicial dels registres i posicions de memòria és el següent:

R1=2454h
R2=5656h
R3=FFFFh
R4=0000h
R5=0002h

Memòria

Adreça	Contingut
00h	0339h
01h	63AFh
02h	EA00h
03h	3304h
04h	7834h
05h	54AAh

 - SUB R1,R2,R5
 - ADDI R3, #1, R1
 - LOAD 3(R5), R1
 - STORE R4, 1(R4)
 - ASR R2, R7
- Indiqueu quina operació realitza la següent seqüència d'instruccions:

D024h
C864h
D844h
- La instrucció *SWAP a, b* és present en alguns llenguatges d'alt nivell. Aquesta instrucció intercanvia els valors de les variables *a* i *b*. Escriviu en llenguatge màquina de la MR un programa que es comporti com aquesta instrucció, suposant que les variables *a* i *b* són:
 - la variable *a* en la posició de memòria 60h i la variable *b* en el registre R2 de la Unitat de Procés.
 - la variable *a* en la posició de memòria 25h i la variable *b* en la posició 1Dh.
- Escriviu en llenguatge màquina de la MR un programa que compti el nombre de bits que valen 1 en un nombre NUM emmagatzemat en memòria. Supposeu que el nombre NUM és a l'adreça 03h i que les instruccions s'emmagatzemen a partir de l'adreça 04h.

R3 := R3 + 1;

fmentre

Problemes de la Unitat de Procès

15. Indiqueu com afectaria als següents elements de la MR el fet d'augmentar de 8 a 16 el nombre de registres del Banc de Registres.
- Grandària de la instrucció.
 - Amplària de la memòria.
 - Busos interns de dades de la Unitat de Procès.
 - Registres interns de la Unitat de Procès.
16. Indiqueu com afectaria als següents elements de la MR el fet d'augmentar la grandària de la memòria fins a tenir una memòria de 64 K paraules en comptes de 256 paraules.
- Registre Comptador de Programa (PC).
 - Registre d'adreces (R@).
 - Grandària de la instrucció (Registre IR) i amplària de la memòria.
 - Amplària del Banc de Registres i busos interns.
17. Indiqueu quines modificacions s'haurien de fer als següents elements de la MR si es vol treballar amb nombres a complement a dos de 32 bits, en comptes de fer-ho amb nombres de 16 bits.
- La UAL
 - La memòria
 - El Banc de Registres
 - Els busos de dades interns de la Unitat de Procès
 - Els registres interns de la Unitat de Procès
18. Realitzeu el disseny de la Unitat d'Avaluació de la Condició de Salt.
19. Redissenyeu la UAL per tal que, a més a més dels indicadors de condició Z i N, també calculi els indicadors:
- V: sobreiximent per a enters
 - C: sobreiximent per a naturals
20. Contesteu els apartats següents:
- Redissenyeu la UAL perquè un únic bloc combinacional realitzi les operacions de suma i de resta que necessiten les instruccions ADD, SUB, ADDI i SUBI.
 - Cal modificar la codificació de les instruccions?
21. Volem substituir la instrucció ASR de la MR per una instrucció que compari dos operands font (*CMP R1I, R1I2*) i modifiqui l'indicador de condició Z tal com s'indica a continuació:
- Z=1 si els dos operands són iguals
 - Z=0 si els dos operands són diferents
- L'indicador de condició N pot prendre qualsevol valor.
- Dissenyeu la nova UAL i redissenyeu la circuiteria que envolta la UAL.
 - Quines implicacions té el fet d'intercanviar la instrucció ASR per la instrucció CMP en el format de les instruccions?
22. Suposant que el temps de resposta dels diferents elements de la Unitat de Procès és el següent:
- 10 ns. per a cada multiplexor i descodificador
 - 50 ns. per al sumador d'adreces i l'incrementador connectat a l'entrada del PC
 - 40 ns. per al bloc que realitza l'avaluació de la Condició de Salt
 - 20 ns. per a llegir un registre del Banc de Registres
 - 100 ns. el temps de resposta de la UAL
 - 100 ns. el temps necessari per a llegir o escriure una dada a la memòria RAM

Contesteu les preguntes següents, sense tenir en compte el temps que pugui tardar a reaccionar la Unitat de Control.

- Calculeu el temps necessari per a executar cadascuna de les fases de cada instrucció (cada estat del graf d'estats reduït).
- Indiqueu la freqüència màxima de relotge a la qual pot funcionar la MR.

23. Suposant que s'inclou un registre RMEM entre el multiplexor SELDAT i la memòria, que no afecta el camí entre la sortida de la memòria i el registre IR, contesteu els apartats següents:
- Dissenyeu el nou graf d'estats per a executar les instruccions de la MR de forma que s'optimitzi el nombre d'estats necessari per a executar la instrucció LOAD. Indiqueu la taula de sortides dels nous estats
 - Contesteu els apartats a) i b) de l'activitat 22 usant el nou graf d'estats.
 - Quina millora representa (en temps d'execució) el fet d'incloure el registre RMEM a la Unitat de Procés?
24. Volem dissenyar una màquina amb el mateix joc d'instruccions que la MR.
- Proposeu una codificació d'instruccions suposant que la màquina:
 - té 16 registres de propòsit general de 32 bits.
 - treballa amb nombres en complement a dos de 32 bits.
 - té una memòria de 2^{22} paraules de 32 bits.Quina és la grandària màxima de l'operand immediat?
 - Quines modificacions s'haurien de fer a la resta de la Unitat de Procés?
 - Com es pot augmentar el nombre d'instruccions aritmètiques fins a 64? Quines repercussions tindria aquest augment sobre la resta de paràmetres de les instruccions?
 - Es podrien afegir més instruccions de salt sense modificar la resta de formats de les instruccions? Quantes i quines suggeriríeu, suposant que a la Unitat de Procés s'afegissin els indicadors de condició C (transport) i V (sobreeiximent)?
25. Contesteu les preguntes següents:
- Dissenyeu un Banc de Registres que tingui dos ports de lectura. El fet que un Banc de Registres tingui dos ports de lectura implica que es poden llegir dos registres al mateix temps.
 - Quins canvis comportaria en el disseny de la Unitat de Procés de la MR substituir el seu Banc de Registres pel que es proposa en l'apartat a)? Hauria de canviar el format de les instruccions?
26. Volem afegir al joc d'instruccions de la MR una instrucció aritmètica que multipliqui per dos un operand font emmagatzemat en un registre i deixi el resultat en un registre destinació (*MULDOS Rf, Rd*). Volem una instrucció específica (aquesta operació es pot fer amb la instrucció *ADD Rf,Rf,Rd*, però no volem aquesta solució).
- Quines implicacions té per al format de les instruccions el fet d'afegir aquesta instrucció?
 - Dissenyeu la nova UAL i redissenyeu la circuiteria que envolta la UAL.

Problemes de la Unitat de Control

27. Donat el programa següent, escrit en llenguatge ensamblador de la MR:

```
00h BR 03h
01h SUB R0, R0, R0
02h BR 05h
03h ADD R0, R0, R0
04h BGE 01h
```

Contesteu les preguntes següents:

- En quin ordre s'executen les instruccions?
- Quants cicles tarda a executar-se el programa (suposant el graf més simplificat per a la Unitat de Control)?
- Indiqueu els diferents valors que pren el PC en cada una de les fases d'execució de les instruccions durant l'execució del programa.
- Compareu la seqüència d'adreces emmagatzemades al PC amb la seqüència d'adreces corresponent a l'execució d'instruccions. Es corresponen totes dues seqüències? Justifiqueu la resposta.

28. Donat el bucle següent, escrit en llenguatge ensamblador de la MR:

```
00h SUB R3, R2, R0
01h BL 08h
02h SUB R4, R2, R0
03h BNE 06h
04h ADDI R2, #1, R2
05h BR 07h
06h SUB R4, #2, R4
07h BG 00h
```

Suposant que inicialment $R3 = 100_{10}$ i $R2 = 49_{10}$, contesteu els apartats següents:

- Descriviu què fa el cos del bucle.
- Quants cops s'executa el bucle?

29. Que indiquen a la Unitat de Procés de la MR aquests senyals de control?

- $Ld_PC = 1$, $Ld_IR = 1$ i $PC/@ = 0$
- $Ld_PC = 1$, $Ld_IR = 1$ i $PC/@ = 1$
- $Ld_RA = 1$ i $CRf = 1$
- $ERd = 1$
- $Ld_RZ = 1$, $Ld_RN = 1$, $ERd = 1$, $OPERAR = 1$ i $CRf = 2$

30. Dissenyu la Unitat de Control de la MR com un sistema lògic seqüencial, la funció d'estats del qual es realitza de forma mínima a tres nivells, i la funció de sortida del qual es realitza amb una memòria ROM.

31. Contesteu els apartats següents:

- Per què s'ha de realitzar la descodificació de la instrucció?
- Per què s'intenta reduir el nombre d'estats de la Unitat de Control de la MR?

32. Si el Banc de Registres de la Unitat de Procés de la MR tingués dos ports de lectura (vegeu l'activitat 25):

- Podria ser més curt el temps d'execució d'alguna instrucció?
- Milloraria el rendiment global de la MR durant l'execució d'instruccions?

33. Si la UAL de la MR, a més a més dels indicadors de condició Z i N, també calculés els indicadors

- V: sobreeximent per a enters
- C: sobreeximent per a naturals

seria interessant tenir més instruccions de salt.

- Quines instruccions de salt es podrien afegir?

b) Caldria canviar el format de les instruccions?

Problemes bàsics de Llenguatge ensamblador

34. Tradueix el programa següent, escrit en un llenguatge d'alt nivell, a llenguatge ensamblador de la MR i posteriorment a llenguatge màquina, suposant que:

- s'emmagatzema a partir de l'adreça 00h,
- les variables a i b són a les posicions de memòria 00h i 01h respectivament,
- la primera instrucció executable del programa és a l'adreça 02h,
- la instrucció *swap* a, b intercanvia els valors de les variables a i b .

```

programa exemple1;
var a, b: enter;
a := 13;
b := 16;
mentre (a > 10) fer
    a := a - 1;
    b := b + 2;
fmentre;
si (a < b)      llavors swap (a, b)
                si no b := a - 1

fsi;
fprograma.

```

35. Tradueix el programa següent, escrit en un llenguatge d'alt nivell, a llenguatge ensamblador de la MR i posteriorment a llenguatge màquina, suposant que s'emmagatzema a partir de l'adreça 00h.

```

programa exemple2;
var a, b, c : enter;
a := 13;
b := 1;
c := 0;
mentre (a > 10) o (b < 20) fer
    si (a < 11) i (b > 18) llavors c := c + 3 fsi;
    a := a - b;
fmentre;
fprograma.

```

36. El programa següent calcula el màxim comú divisor de dos nombres a i b segons l'algorisme de residus d'Euclides:

```

programa exemple3;
var a = 5, b = 15, mcd: enter;
mentre (a <> b) fer
    si (a > b)      llavors a := a - b
                    si no b := b - a

fsi;
fmentre;
mcd := a;
fprograma.

```

a) Feu-ne la traducció a llenguatge ensamblador de la MR.

b) Tradueix-lo a llenguatge màquina, expressant-lo tant en binari com en hexadecimal, suposant que:

- s'emmagatzema a partir de l'adreça 0Fh,
- les variables a i b s'emmagatzemen a les posicions 0Fh i 10h respectivament,
- la variable *mcd* s'emmagatzema a la posició 11h,
- la primera instrucció executable del programa és a l'adreça 12h.

Problemes avançats de Llenguatge ensamblador

37. Escriviu en un llenguatge d'alt nivell un programa que calculi el quadrat d'un nombre positiu a mitjançant la suma acumulada a vegades. Feu-ne la traducció a llenguatge ensamblador de la MR i posteriorment a llenguatge màquina, suposant que s'emmagatzema a partir de l'adreça 00h.
38. L'algorisme següent, descrit en un llenguatge d'alt nivell, calcula el factorial d'un nombre a . Feu-ne la traducció a llenguatge ensamblador de la MR i posteriorment a llenguatge màquina, suposant que s'emmagatzema a partir de l'adreça 00h. Indiqueu el contingut de la taula de símbols.

```

Programa FACTORIAL;
var n, ind1, ind2, asumir, acum, fact: enter;
n:= 9;
ind1:= n;
asumar := n;
acum:= 0;
mentre ( ind1 > 2) fer
    ind2 := ind1 - 1;
    mentre ( ind2 > 0 ) fer
        acum := acum + asumir;
        ind2 := ind2 - 1;
    fmentre;
    asumir := acum;
    acum := 0;
    ind1 := ind1 - 1;
fmentre;
fact := asumir;
fprograma.

```

39. Escriviu en un llenguatge d'alt nivell un programa que calculi el nombre de lletres "a" d'una frase acabada en punt. Traduïu el programa a llenguatge ensamblador i llenguatge màquina de la MR, suposant que:
- El programa es troba emmagatzemat a partir de l'adreça 80h.
 - La frase es troba emmagatzemada a partir de l'adreça 02h en posicions consecutives de memòria.
 - Cada lletra ocupa una posició de memòria i està codificada utilitzant el codi ASCII. En aquest sistema de codificació, cada símbol es codifica amb 7 bits; en aquest cas, en els 7 bits de menys pes de la posició de memòria que ocupa. Els 9 bits de més pes són a 0. La lletra "a" es codifica mitjançant el nombre 61h, i el caràcter "." mitjançant el nombre 2Eh.
 - Els valors 61h i 2Eh es poden emmagatzemar a les adreces 00h i 01h. La frase mai no té més de 125 lletres, i sempre acaba en ".".
40. Escriviu en un llenguatge d'alt nivell un programa que calculi la suma dels elements d'un vector al mateix temps que busca els elements màxim i mínim. Feu la traducció a llenguatge ensamblador de la MR i posteriorment a llenguatge màquina, suposant que les dades s'emmagatzemen a partir de l'adreça 00h i les instruccions a partir de l'adreça 3Ah.
41. Escriviu en un llenguatge d'alt nivell un programa que sumi, element a element, dos vectors A i B, deixant el resultat en un tercer vector C ($C[i] := A[i] + B[i]$). La grandària dels vectors és de 10 elements. Traduïu el programa a llenguatge ensamblador de la MR i posteriorment a llenguatge màquina, suposant que els vectors s'emmagatzemen a partir de l'adreça de memòria 00h i el programa s'emmagatzema a partir de l'adreça 80h.
42. Escriviu en llenguatge ensamblador de la MR les macros que permeten d'executar les instruccions següents, pertanyents als jocs d'instruccions d'altres computadors:
- | | |
|--------------|---|
| a) clr Rd | Posa un 0 al registre destinació |
| b) clr A(Ri) | Posa un 0 a la posició de memòria M[A+(Ri)] |
| c) inc Rd | Incrementa en una unitat el valor del registre Rd |
| d) dec Rd | Decrementa en una unitat el valor del registre Rd |

- e) `add3 Rf1, Rf2, Rd` Suma el contingut de 3 registres del Banc de Registres i deixa el resultat en un d'ells (Rd)
- f) `asl Rd` Desplaça un bit a l'esquerra el registre Rd
- g) `mov Rf, Rd` Copieu el valor del registre Rf en el registre Rd
- h) `mov Rf, A(Ri)` Copia el valor del registre Rf en la posició de memòria $M[A+(Ri)]$
- i) `mov A(Ri), B(Rj)` Copia el valor de la posició de memòria $M[A+(Ri)]$ en la posició de memòria $M[B+(Rj)]$
- j) `swap Rf1, Rf2` Intercanvia els valors dels registres Rf1 i Rf2
- k) `swap A(Ri), B(Rj)` Intercanvia els valors de les posicions de memòria $M[A+(Ri)]$ i $M[B+(Rj)]$
- l) `asrn #n, Rd` Desplaça n bits a la dreta el registre Rd. El nombre n ha d'ésser més gran que 0 i més petit que 16
- m) `asln #n, Rd` Desplaça n bits a l'esquerra el registre Rd. El nombre n ha d'ésser més gran que 0 i més petit que 16
- n) `inc A(Ri)` Incrementa el contingut de la posició de memòria $M[A+(Ri)]$

43. Escriviu en un llenguatge d'alt nivell un programa que calculi la suma dels elements d'un vector V que compleixen el fet d'ésser més grans que un cert valor MÍNIM i més petits que un cert valor MÀXIM ($MÍNIM < V[i] < MÀXIM$). Feu-ne la traducció a llenguatge ensamblador de la MR i posteriorment a llenguatge màquina, suposant que el programa s'emmagatzema a partir de l'adreça 3Ah.
44. Tradueix a llenguatge ensamblador de la MR el programa següent, escrit en un llenguatge d'alt nivell. El programa calcula el factorial d'un nombre "a" (en l'exemple, a=8):

```

programa FACTORIAL;
var a, cont, factorial: enter;
a:= 8;
cont := 1;
factorial := 1;
mentre ( cont < A ) fer
    cont := cont + 1;
    factorial := factorial * cont;
fmentre;
fprograma.

```

- a) Dissenyu i useu la macro `mul $1, $2, $3`, que multiplica el contingut de dos registres (\$1, \$2) i deixa el resultat en un tercer registre (\$3). Els tres registres han d'ésser diferents.
- b) Feu el preassemblatge (expansió de macros) i el postassemblatge (creació de la taula de símbols i generació de codi) del programa, suposant que el programa s'emmagatzema a partir de l'adreça de memòria 00h.
45. Donat el programa següent, escrit en llenguatge ensamblador de la MR:

```

N:      .DW 4
RESUL:.RW 1
        .DEF cmp $1, $2
            SUB $1, $2, R0
        .ENDDEF
        .DEF cmp $1, $i2
            SUBI $1, $i2, R0
        .ENDDEF
        .DEF clr $1
            ADD R0, R0, $1
        .ENDDEF
        .DEF mov $1, $2
            ADD R0, $1, $2
        .ENDDEF
        .DEF mov $i1, $2
            ADDI R0, $i1, $2
        .ENDDEF
        .DEF mov $d1, $2
            LOAD $d1, $2
        .ENDDEF
        .DEF mov $1, $d2
            STORE $1, $d2

```

```
.ENDDEF
.DEF inc $1
    ADDI $1, #1. $1
.ENDDEF
.DEF dec $1
    SUBI $1, #1. $1
.ENDDEF
.BEGIN INI
INI:  MOV N(R0), R1
      MOV R1, R3
      clr R4
M1:   cmp R1, #2
      BLE FM1
      SUBI R1, #1, R2
M2:   cmp R2, #0
      BLE FM2
      ADD R4, R3, R4
      dec R2
      BR M2
FM2:  mov R4, R3
      clr R4
      dec R1
      BR M1
FM1:  mov R3, RESULT(R0)
.END
```

- Realitzeu-ne el preassemblatge (expansió de macros).
- Realitzeu-ne el postassemblatge (generació de la taula de símbols i traducció a llenguatge màquina) suposant que el programa s'emmagatzema a partir de la posició 00h de memòria.
- Traduïu el programa a alt nivell, indicant quines operacions realitza. Per a fer això, intenteu analitzar quines operacions executa cada macro.