PAC1

Estructura de computadors

Grau en Enginyeria Informàtica

2013 S2

Estudis d'informàtica, multimèdia i comunicació

EIMT.UOC.EDU

UOC
Universitat Oberta de Catalunya

Presentació

La present PAC1 conté 5 preguntes i representa el 50% de la nota de l'avaluació contínua.

Com podreu veure, els exercicis són molt semblats als quals heu fet durant aquests dies, en els quals a més heu pogut donar les solucions, comentar-les i plantejar dubtes en el fòrum. Aquesta PAC és **individual**, **avaluable** i per tant no pot comentar-se.

Competències

Les competències específiques que persegueix la PAC1 són:

- [13] Capacitat per identificar els elements de l'estructura i els principis de funcionament d'un ordinador.
- [14] Capacitat per analitzar l'arquitectura i organització dels sistemes i aplicacions informàtics en xarxa.
- [15] Conèixer les tecnologies de comunicacions actuals i emergents i saber-les aplicar convenientment per dissenyar i desenvolupar solucions basades en sistemes i tecnologies de la informació.

Objectius

Els objectius de la següent PAC són:

- Conèixer el joc d'instruccions de la màquina CISCA.
- Conèixer els modes d'adreçament de la màquina CISCA.
- Traduir petits programes a assemblador.
- Comprendre la codificació interna de les instruccions d'assemblador.
- Descriure les micro-operacions involucrades en l'execució d'una instrucció d'assemblador.

Enunciat

Respondre cada pregunta o apartat en el requadre corresponent.

Recursos

Podeu consultar els recursos disponibles a l'aula, però no fer ús del fòrum.

Criteris de valoració

La **puntuació** de cada pregunta i els **criteris d'avaluació** els trobareu a cada pregunta.

Format i data de lliurament

- La PAC1 podeu lliurar-la a l'apartat de **lliurament d'activitats** amb el nom **cognom1_cognom2_nom_PAC1 (pdf / odt / doc / docx).**
- La data **límit** de lliurament és el **11/04/2014.**

Enunciat

Pregunta 1 (2 punts)

Suposeu que l'estat inicial del computador (el valor que contenen els registres, posicions de memòria i bits de resultat just abans de començar l'execució de cada fragment de codi, de cada apartat) és el següent:

- Registres: Ri = 16*i per a i = 0, 1, …,15.

Exemple: R0=0, R1=16, R2=32, … (valors en decimal)

- Memòria: M(i)=(i+16) per a i = 0, 4, 8,...

La notació M(i) es refereix a la paraula de 4 bytes emmagatzemada en les adreces i, i+1, i+2. i+3 en little endian (i ha de ser múltiple de 4).

Exemples: M(00001000h) = 00001010h
M(0000FFF8h) = 00010008h

- Bits de resultat del registre d'estat: Z=0, S=0, C=0, V=0

- Registres especials: suposem que el PC apunta a l'inici del fragment de codi de cada apartat.

Quin serà l'estat del computador després d'executar cadascun dels següents fragments de codi? Indiqueu solament el contingut (**en hexadecimal**) dels registres i posicions de memòria que s'hagin modificat com a resultat de l'execució del codi. Indiqueu el valor final de tots els bits de resultat. (No us demanem que indiqueu el valor del PC després d'executar el codi i per això no us hem donat el valor inicial del PC, on comença cada fragment de codi).

Suposeu que l'adreça simbòlica `A` val 100h.

a)

```
        ADD  R4, R2
        SUB  R0, R4
        JLE  GO
        MOV  R5, 0
        JMP  FIN
GO:     MOV  R5, 20H
FIN:
```

Solució:

R4 = R4+R2 = 40H+20H =60H
R0 = 0 – 60H = FFFFFFA0H
R5 = 20H

Z=0, S=1, C=1, V=0

b)

```
MOV R5, [4444CCCCH]
SAL R5, 1
```

Solució:

R5 = 4444CCDCh
R5 = 888999B8h

Z=0, S=1, V=1, C=0

c)

```
        MOV  R0, [A]
        MOV  R1, 4
INI:    SAL  R0, 2
        OR   R0, [A]
        SAR  R1, 1
        JNE  INI
FIN:    MOV [A],R0
```

Solució:

```
R0 = [100H] = 110H
R1 = 4
R0 = 110 << 2 = 00000440H
R0 = 00000440H OR 110h = 00000550H
R1 = 4 >> 1 = 2

R0 = 00000550H << 2 = 00001540H
R0 = 00001540H OR 110h = 00001550H
R1 = 2 >> 1 = 1

R0 = 00001550H << 2 = 0005540H
R0 = 00005540H OR 110h = 00005550H
R1 = 1 >> 1 = 0

[A] = [100H] = 00005550H
Z=1, S=0, V=0, C=0
```

Criteris de valoració. Els apartats a) i b) valen 0,5 punts cadascun i el c) val 1 punt. La valoració de cada apartat és del 100% si no hi ha cap error en la solució de l'apartat, és del 50% si el contingut de les posicions de memòria i registres modificats és correcte però hi ha algun error en el contingut de un o varis dels bits de resultat, i és del 0% si hi ha algun error en alguna posició de memòria o registre modificat.

Pregunta 2 (1 punt)

Escriu un petit codi d'assemblador que implementi el següent codi d'alt nivell utilitzant el mode d'adreçament a memòria:

Si (A>B) o (C>D) llavors A= B+D

Solució:

```
      MOV R0,[A]
      CMP R0,[B]
      JG SUMA
      MOV R0,[C]
      CMP R0,[D]
      JLE FINAL
SUMA:MOV R0,[B]
      ADD R0,[D]
      MOV [A],R0
FINAL:
```

Criteris de valoració. 1 punt si no hi ha cap error a la solució. 0,5 si modificant una sola instrucció del codi aquest fora correcte, i és del 0 si perquè el codi fos correcte calgués modificar més de una instrucció.

Pregunta 3 (3.5 punts)

A cada apartat us presentem un codi escrit en el llenguatge d'alt nivell C i una traducció parcial a llenguatge assemblador de la màquina CISCA. També s'expliquen les variables usades en el codi i alguna indicació que us ajudi a comprendre la traducció.

Completeu la traducció a llenguatge assemblador CISCA que proposem a cada apartat.

a) (2 punts)

En el codi de alt nivell `D` és una variable de tipus vector d'enters de 32 bits que a el programa assemblador es troba emmagatzemada a partir de l'adreça simbòlica `D` (el primer element de D , `D[0]`, en l'adreça simbòlica `D`, el següent, `D[1]`, en l'adreça `D+4`, etc.). Aquest vector té 100 elements. El mode d'adreçament d'accés al vector ha de ser indexat.

```
k=0;
for (i=0; i<N; i++)
 for (j=0; j<N; j++)
        if (D[k] > D[i] + D[j])
        {
        D[k] = D[i] + D[j];
        k++;
        }
```

```
      MOV R1,0 ;i
      MOV R2,0 ;j
      MOV R3,0 ;k
F1:
      CMP R1, 400
      JGE FINAL
F2:   CMP R2,400
      JGE FF1

      MOV R4, [D+R1]
      ADD R4, [D+R2]
      CMP [D+R3], R4
      JLE FF2
      MOV [D+R3], R4
      ADD R3, 4

FF2:  ADD R2, 4
      JMP F2
FF1:  ADD R1, 4
      MOV R2, 0
      JMP F1
FINAL:
```

Criteris de valoració. Si tot està correcte s'obté un 2. Si es té solament una línia incorrecta s'obté un 1, si són dues línies incorrectes s'obté un 0.5 i si són tres o més les línies incorrectes s'obté un 0.

b) (1.5 punts)

Us presentem un codi en alt nivell i una traducció parcial a llenguatge assemblador de la màquina CISCA. En el codi en alt nivell, A és un vector d'enters i MAX és un enter. A es troba emmagatzemat a partir de l'adreça simbòlica A (el primer element d' A, A[0], en l'adreça simbòlica A, el següent, A[1], en l'adreça A+4, etc).

Completeu la traducció proposada de llenguatge assemblador de manera que l'execució correspongui al que es demana en alt nivell. En aquest codi, l'índex "i" s'implementa amb R0.

Utilitzeu el mode d'adreçament indexat per a l'accés al vector.

```
i:= 0;
do {
  if(A[i]>i) i=A[i];
  else i++;
} while i<MAX
```

```
     MOV R0, 0
     MOV R1, 0
DO:  MOV R2, [A + R1]
     CMP R2,R0
     JLE ELSE
     MOV R0,[A + R1] ; o R2
     MOV R1, R0
     MUL R1,4
ELSE:ADD R0, 1
     ADD R1, 4
     CMP R0, [MAX]
     JL DO
```

Criteris de valoració. Si tot està correcte s'obté un 1.5. Si es té solament una línia incorrecta s'obté un 1, si són dues línies incorrectes s'obté un 0.5 i si són tres o més les línies incorrectes s'obté un 0.

Pregunta 4 (2 punts)

Donat el següent fragment de codi de un programa en llenguatge assemblador del CISCA:

```
        MOV R0, 0
        MOV R1, R0
A:      ADD R0, [V+R1]
        ADD R1, 8
        CMP R1, 20
        JLE A
```

Traduïu-lo a llenguatge màquina i expresseu-lo en la següent taula. Suposeu que la primera instrucció del codi s'assembla a partir de l'adreça **3FC0h** (que és el valor del PC abans de començar l'execució del fragment de codi). Suposeu que l'adreça simbòlica `V` val **80h**. En la següent taula useu una fila per codificar cada instrucció. Si suposem que la instrucció comença en l'adreça @, el valor Bk de cadascun dels bytes de la instrucció amb adreces @+k per a k=0, 1,... s'ha d'indicar en la taula en hexadecimal en la columna corresponent (recordeu que els camps que codifiquen un desplaçament en 2 bytes o un immediat o una adreça en 4 bytes ho fan en format little endian, això cal tenir-ho en compte escrivint els bytes de menor pes, d'adreça més petita, a l'esquerra i els de major pes, adreça major, a la dreta). Completeu també la columna 'Adreça' que indica per a cada fila l'adreça de memòria del byte B0 de la instrucció que es codifica en aquesta fila de la taula.

		Bk para k=0..10										
Dirección	**Ensamblador**	0	1	2	3	4	5	6	7	8	9	10
00003FC0	MOV R0, 0	10	10	00	00	00	00	00				
00003FC7	MOV R1, R0	10	11	10								
00003FCA	ADD R0, [V+R1]	20	10	51	80	00	00	00				
00003FD1	ADD R1, 8	20	11	00	08	00	00	00				
00003FD8	CMP R1, 20	26	11	00	14	00	00	00				
00003FDF	JLE A	44	60	E7	FF							
00003FE3												

Criteris de valoració. En aquesta pregunta els errors van restant punts. Cada instrucció assemblada incorrectament resta 0.5 punts. Una instrucció està incorrectament assemblada si no s'escriu el valor o s'escriu un valor incorrecte de un o varis dels dígits hexadecimal que codifiquen la instrucció en llenguatge màquina. A més es resta 0.5 punts si hi ha algun error en la columna que indica les adreces de memòria en les quals comença cada instrucció.

Pregunta 5 (1.5 punts)

El *cicle d'execució* de una instrucció es divideix en 3 fases principals
1) Lectura de la instrucció
2) Lectura dels operands font
3) Execució de la instrucció i emmagatzematge de l'operant destinació

Donar la seqüència de micro-operacions que cal executar en cada fase per a les següents instruccions del codi codificat en la pregunta anterior.

`ADD R0, [V+R1]`

Fase	Micro-operacions

1	`(MAR=00003FCAh) <- (PC=00003FCAh), READ ;Contingut del PC al`
	`;registre MAR.`
	`(MBR=00000080511020h) <- Memòria ;Llegim la instrucció`
	`(PC=00003FD1h) <- (PC=00003FCA+7 ;Incrementem el PC en 7`
	`;unitats (mida de la instrucció).`
	`(IR=00000080511020h) <- (MBR=00000080511020h) ;Carreguem la`
	`;instrucció en el registre IR.`

2	`MAR <- IR(Adreça V)=00000080h + R1, READ`
	`MBR <- Memòria`
	`;L'operand (R0) és un registre. No s'ha de fer res.`

3	`R0 <- R0 + MBR ;Executem la instrucció`

CMP R1, 20

Fase	Micro-operacions
1	(MAR=00003FD8h) <- (PC=00003FD8h), READ ;Contingut del PC al ;registre MAR. (MBR=00000014001126h) <- Memòria ;Llegim la instrucció (PC=00003FDFh) <- (PC=00003FD8h+7 ;Incrementem el PC en 7 ;unitats (mida de la instrucció). (IR=00000014001126h) <- (MBR=00000014001126h) ;Carreguem la ;instrucció en el registre IR.
2	L'operand font (20) és un immediat i està en la mateixa instrucció. L'operand destí (R1) és un registre. No s'ha de fer res.
3	R1-20; executem la instrucció implícita de CMP (destí – font) actualitzant els bits de resultat.

Criteris de valoració. Si tot està correcte s'obté 1.5 punts. Cada instrucció és independent i val 0.75. Es resta 0.25 per cada fallada dins de la mateixa instrucció.