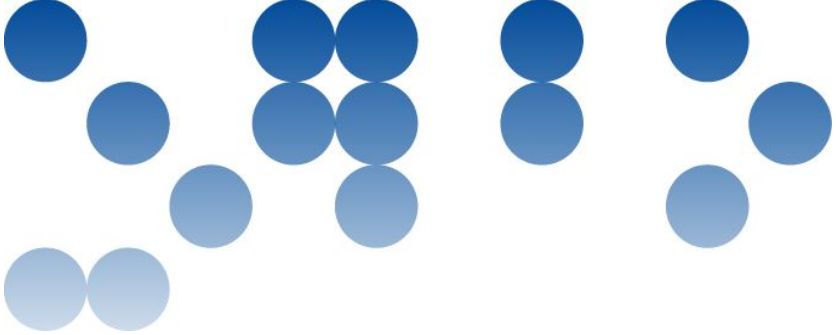

PAC1

Estructura de Computadors

Grau en Enginyeria Informàtica

feb19-jun19

Estudis d'Informàtica, Multimèdia i Telecomunicació

EIMT.UOC.EDU

Universitat Oberta de Catalunya

Presentació

La present PAC1 conté 4 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 (en format pdf).**
- La data **límit** de lliurament és el **15/03/2019.**

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:

R0= 400h R1= 300h R2= 200h R3= 100h R4= 500h	M(00000100h)=FFFFF000h M(00000200h)=0000FFFFh M(00000300h)=11110000h M(00000400h)=00001111h M(00000500h)=F000000Fh M(00000600h)=00000500h

- 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 es 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 00000600h

Important: deixeu clarament indicat (**preferiblement ressaltat o d'altre color**) a cada apartat **el valor final** dels registres i adreces de memòria modificades així com els bits d'estat.

a)

```
MOV   R0, [R1]
SUB   R0, [R2]
MOV   R3, R0
```

Solució:

R0:= [300h]= 11110000h
R0:= 11110000h - [200h]= 11110000h-0000FFFFh= 11100001h
R3:= 11100001h

==========
R0= 11100001h
R3= 11100001h

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

b)

```
NOT [R2]
MOV R3,[A]
SUB R2,R3
```

Solució:

[R2]:= NOT(00000200h)= NOT 0000FFFFh = FFFF0000h
R3:= 500h
R2= 200h - 500h= FFFFFD00h

==========
[200h]= FFFF0000h
R3= 500h
R2= FFFFFD00h
Z= 0 , S= 1 , C= 1 , V= 0

c)

```
       MOV  R1, [R1]
PLUS:  CMP  R2,R0
       JG   END
       SUB  R1, [R2]
       ADD  R2, 100h
       JMP  PLUS
END:
```

```
Solució:

     R1:= 11110000h:

     // R2= 200h
     R1:= 11110000h - 0000FFFFh = 11100001h
     R2:= 300h
     Z=0 , S=0, C=0, V= 0

     // R2= 300h
     R1:= 11100001h - 11110000h = FFFF0001h
     R2:= 400h
     Z=0 , S=0, C=0, V= 0

     // R2= 400h
     R1:=  FFFF0001h - 00001111h = FFFEEEF0h
     R2:= 500h
     Z=0 , S=0, C=0, V= 0

     // R2= 500h

     ======================
     R1:= FFFEEEF0h
     R2:= 500h

     Z=0 , S=0, C=0, V=  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 (2 punts)

En el codi de alt nivell `M` és una variable de tipus vector de 10 elements. Cada element de la matriu és un enter de 32 bits. A el programa assemblador la matriu es troba emmagatzemada a partir de l'adreça simbòlica `M`, en posicions consecutives (`M[0]`, en l'adreça simbòlica `M`, el següent, `M[1]`, en l'adreça `M+4`, etc.).

`SOURCE` és una variable de tipus vector d'una dimensió de 10 elements. Cada element del vector és un enter de 32 bits. Al programa assemblador la matriu es troba emmagatzemada a partir de l'adreça simbòlica `SOURCE` en posicions consecutives (`SOURCE[0]`, en l'adreça simbòlica `SOURCE`, el següent, `SOURCE[1]`, en l'adreça `SOURCE+4`, etc.).

El programa recorre els 2 vectors buscant una coincidència en la mateixa posició de tots dos.

Utilitzarem el registre R1 per implementar la variable de control “i” i R2 per al control de “j”

```
I= 0; J= 0;
WHILE ((I<10) && (J!=1)) {
  IF (M[I]==SOURCE[I]) J:= 1;
  ELSE I:= I+1;
}
```

Emplena els espais de la proposta de programa assemblador que es mostra a continuació per aconseguir el resultat desitjat.

```
Solució:
      MOV R1, 0
      MOV R2, 0
      MOV R3, 0
      MOV R4, M
WI:   CMP R1, 10
      JGE ENDWI
      CMP R2, 1
      JE ENDWI
      MOV R5, [SOURCE+R3]    ;MOV R5, [R4]
      CMP [R4], R5           ;MOV [SOURCE+R3], R5
      JNE CONT
      MOV R2, 1
      JMP WI
CONT: ADD R4, 4
      ADD R3, 4
      INC R1
      JMP WI
ENDWI:
```

Criteris de valoració. 2 punts. Es perden 0,5 punts per cada instrucció incorrecta.

Pregunta 3 (3 punts)

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

```
X:    CMP R1, R3
      JL END
      ADD R2, [R1]
      SUB R1, [V+R3]
      JMP X
END:
```

Traduïu-ho a llenguatge màquina i expresseu-ho en la següent taula. Suposeu que la primera instrucció del codi s'assembla a partir de l'adreça **003FCF00h** (que és el valor del PC abans de començar l'execució del fragment de codi). Suposeu que l'adreça simbòlica V val **00404040h.** 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 per a k=0..10										
Adreça	**Assemblador**	0	1	2	3	4	5	6	7	8	9	10
003FCF00h	CMP R1,R3	26	11	13								
003FCF03h	JL END	43	60	10	00							
003FCF07h	ADD R2, [R1]	20	12	31								
003FCF0Ah	SUB R1,[V+R3]	21	11	53	40	40	40	00				
003FCF11h	JMP X	40	00	00	CF	3F	00					
003FCF17h												

Criteris de valoració. 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 4 (2 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.

SUB R1,[V+R3]

Fase	Micro-operacions
1	(MAR=003FCF0Ah)←(PC=003FCF0Ah) , Read ;posem el contingut del PC en MAR (MBR=00404040531121)←Memòria ;llegim la instrucció (PC=003FCF11h) ← (PC=003FCF0Ah)+7 ;incrementem el PC en 7 unitats IR←(MBR=004040531121h) ;carreguem la instrucció a IR
2	MAR←Contingut de IR adreça V=00404040h + R3, read ;adreça 2n operand al MAR MBR←Memòria ;llegim valor de l'operand
3	R1 = R1 - MBR

JMP X

Fase	Micro-operacions
1	(MAR=003FCF11h)←(PC=003FCF11h), Read ;posem el contingut del PC en MAR (MBR=003FCF000040)←Memòria ;llegim la instrucció (PC=003FCF17h)←(PC=003FCF11h)+6 ;incrementem el PC en 6 unitats IR←(MBR=003FCF000040h) ;carreguem la instrucció a IR
2	Salt a adreça absoluta. No necessitem operacions per obtenir l'operand.
3	(PC=003FCF00h)←IR(Adreça)=003FCF00h

Criteris de valoració. Si tot està correcte s'obtenen 2 punts. Cada instrucció és independent i val 1 punt. Es resta 0.5 per cada fallada dins de la mateixa instrucció.

Pregunta 5 (1 punt)

Respon a les següents preguntes:

Pregunta 1. En què es diferencien els modes d'adreçament «relatiu a registre base» i «relatiu a registre índex»?

En el mode relatiu a registre índex, l'adreça de memòria es troba explícitament en la instrucció i mentre que en el mode relatiu a registre base l'adreça està en el registre.
El mode relatiu a registre base és compacte a nivell de codificació ja que el desplaçament ocupa poc i com hem dit l'adreça està en el registre i no codificada en memòria.
El mode relatiu a a registre índex sol usar-se en accés a estructura com a vectors o taules, mentre que el relatiu a registre base sol utilitzar-se per implementar segmentació.

Pregunta 2. Què són el PC i el SP i per a què s'utilitzen?

El PC (comptador de programa) és un dels registres d'instrucció del processador. Apunta a la següent instrucció a llegir en memòria i s'incrementa automàticament després d'una lectura d'instrucció.
El SP (stack pointer) és un registre que s'actualitza automàticament quan col·loquem o retirem dades de l'estructura dinàmica nomenada Pila (stack) per apuntar a l'última dada introduïda en l'estructura.

Criteris de valoració. Cada pregunta individual val 0,5 punts si està correcta. 0 si no ho està o és incompleta.