PAC1

Estructura de computadors

Programa

2019 s1

Estudis d'informàtica, multimèdia i comunicació

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 compentències 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 microoperacions 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 **11/10/2019.**

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= 000h R1= 100h R2= 200h R3= 300h R4= 400h	M(00000100h)=F0F0F0F0h M(00000200h)=0F0F0F0Fh M(00000300h)=11110000h M(00000400h)=00001111h M(00000500h)=00000500h M(00000600h)=00000300h

- 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).

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

a)

```
MOV  R0, [R3]
ADD  R0, [R4]
ADD  [R1], R0
```

Solució:

R0:= [300h]= 11110000h
R0:= 11110000h + [400h]= 11110000h + 00001111h= 11111111h
[00000100h]= F0F0F0F0h + 11111111h = 02020201h

==========
R0= 11111111h
[00000100h]= 02020201h

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

b)

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

Solución:

[R2]:= NOT 0F0F0F0Fh= F0F0F0F0h
R3:= [00000300h] = 11110000h
R3:= F0F0F0F0h – 11110000h= DFDFF0F0h

==========
[00000200h]= DFDFF0F0
R3= 11110000h

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

c)

```
PLUS:CMP R1, [R1]
     JE END
     ADD R0, [R1]
     ADD R1, 100h
     JMP PLUS
END:
```

Solució:

// R1= 100h
R0:= 00000000h + F0F0F0F0h = F0F0F0F0h
R1:= 200h

// R1= 200h
R0:= F0F0F0F0h + 0F0F0F0Fh = FFFFFFFFh
R1:= 300h

// R1= 300h
R0:= FFFFFFFFh + 11110000h = 1110FFFFh
R1:= 400h

// R1= 400h
R0:= 1110FFFFh + 00001111h = 11111110h
R1:= 500h

// R1= 500h

==========
R1:= 500h
R0:= 11111110h

Z=1 , 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 d'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`, per files i en posicions consecutives (`M[0]`, en l'adreça simbòlica `M`, el següent, `M[1]`, en l'adreça `M+4`, etc.).

El programa busca dins del vector el primer 0 sense passar-se de la dimensió.

Al programa ensamblador R1 adreça la variable de control «i».

```
I= -1;
DO  {
   I:= I+1;
WHILE (M[I]!=0) && (I<9);
}
IF (M[I]==0) M[I]= I;
```

Omple els espais a la proposta que es mostra a continuació per aconseguir el resultat desitjat.

```
Solució:

          MOV R1, -1
          MOV R2, -4
PLUS:     ADD R1, 1
          ADD R2, 4
          CMP [M+R2], 0
          JE ENDDO
          CMP R1 , 9
          JL  PLUS
          JMP END
ENNDO:    MOV [M+R2], R1
END:
```

Criteris de valoració.

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

Pregunta 3 (3 punts)

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

```
        MOV R1,[V+R6]
LOOP:   MUL [V], R2
        ADD R10, [R1]
        CMP R0, 100h
        JL LOOP
```

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 **0003FC00h** (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
0003FC00	MOV R1,[V+R6]	10	11	56	40	40	40	00				
0003FC07	MUL [V], R2	22	20	40	40	40	00	12				
0003FC0E	ADD R10, [R1]	20	1A	31								
0003FC11	CMP R10, 100h	26	1A	00	00	01	00	00				
0003FC18	JL LOOP	43	60	EB	FF							
0003FC1C												

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.

MOV R1,[V+R6]

Fase	Micro-operacions
1	MAR:= 0003FC00, READ; Contingut de PC en el registre MAR. MBR:= 00404040561110; Llegim la instrucció PC:= 0003FC07; Incrementem el PC en 7 unitats (tamany de la instrucció). IR:= MBR; Carreguem la instrucció en el registre IR
2	MAR:= Contingut IR(V) + Contingut IR(R6), READ MBR:= memòria
3	R1:= MBR

CMP R10, 100h

Fase	Micro-operacions
1	MAR:= 0003FC11, READ; Contingut del PC en MAR MBR:= 00000100001A26; llegim la instrucció PC:= 0003FC18h; incrementem el PC en 7 unitats IR:= MBR; Carreguem la instrucció en IR
2	No necessitem operacions per obtenir l'operand
3	R10-RI(100h); executem la instrucció implícita de CMP (destí – font) i modifiquem els bits d'estat.

Criteris de valoració. Si tot està correcte s'obté 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. Quina diferència hi ha entre els bits d'estat C i V?

El bit de carry C és el bit de resultat que s'activa si en una operació de suo o resta de naturals ens portem una. En operacions amb nombres amb signe, el bit equivalent és el overflow V, que indica que el resultat no pot ser representat en el format que estem utilitzant (complement a 2).

Pregunta 2. En què consisteix l'adreçament implícit? Pots posar algun exemple d'instrucció amb aquest tipus d'adreçament?

En l'adreçament implícit, la instrucció no conté informació sobre la localització de l'operant perquè aquest es troba en un lloc predeterminat, i queda especificat de manera implícita en el codi d'operació.

Un exemple serien les instruccions PUSH/POP que treballen amb la pila i utilitzen SP com a registre per adreçar l'operand.

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