***Pràctica 1***

**Objectius**

Els objectius d’aquesta primera practica son familiaritzar-se amb el programa Ripes i començar a entendre el codi per poder fer diferents programes i implementar funcions en el RSCP i el R5SP, a més d’entendre com funciona la memòria on es guarda la informació.

**Informe:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente-1.

-2.

**Treball a fer a clase**

**a)**

El programa guarda el resultat de a2 que és 18 en dues posicions més que zz ja que al punter a0 se li suma 4 quan apunta a zz(0x1000000c), per tant augmenta 1 posició en memòria a (0x10000010) i en la sentencia “final: sw a2,4(a0)” es carrega el contingut de a2 en el punter a0 mes 4, per tant el 18 es guarda en (0x10000014).

**b)**

El programa comença amb les variables xx, yy, oo i zz que reserven en memòria una paraula de 32 bits amb valors numèrics, després s’inicialitzen tres variables a1 i a2 a 0 i a7 a el valor de a1 que es 0 menys 4, per tant -4. Es fa un loop que es repetirà fins que a7 sigui igual a 0, i com que cada vegada a1 augmenta en 1, el loop es repetirà 4 vegades i sortirà al final: on es guardarà el valor de a2 en 4(a0). La variable a2 va sumant els valors que va emmagatzemant a0, a0 comença apuntant a xx que conte el valor 3, i en cada loop augmenta la seva direcció en 4 per tant passarà a emmagatzemant el contingut de yy, després oo i finalment zz, d’aquesta manera a2 al final del programa contindrà un 18 que es el que es guardarà en 4(a0).

**c)**

La sentencia de control de flux es el loop for.

**d)**

La funció de a1 es anar augmentant el seu valor en 1 fins arribar a 4, que s’emmagatzemarà en a7 i farà que el loop s’aturi.

La funció de a0 es guardar el valor de les paraules xx yy oo zz per a que es sumin en a2 i finalment guardar en resultat de a2.

-3.

**e)**

La relació entre el source code i el executable code és una traducció, el codi que escriu el programador, es a dir el source code, es tradueix a llenguatge maquina en el executable code.

**f)**

El PC comença en 0 ja que es un punter que apunta a la següent instrucció a realitzar, per tant com la primera instrucció esta en la adreça de memòria 0 el PC començarà apuntant a 0. Quan s’executi la primera instrucció el PC passarà a apuntar a la següent instrucció que estarà en la posició 4, aquest valor vindrà donat pel sumador que augmenta en 4 el valor al que apunta el PC. Això ho seguirà fent fins que arribi al loop, en aquest cas haurà de tornar cap a instruccions que estan enrere. En aquest moment l’ALU donarà el valor del punter al que apuntarà el PC per executar instruccions anteriors.

**g)**

la a0, xx

assigna l’adreça de xx en a0.

a0 = 0x10000000 (conté un 3)

sub a1,a1,a1

resta el valor de a1 menys el valor de a1 i guarda el resultat en a1

a1 = 0

add a2,zero,zero

suma 0 mes 0 i guarda el resultat en a2

a2 = 0

addi a7,a1,-4

suma el valor de a1 mes l’immediat -4 i guarda el resultat en a7

a7 = -4

beqz a7,final

si a7 es igual a 0 el programa saltarà a l’etiqueta final

a7 = -4 per tant no farà el salt

lw a3,0(a0)

carrega el valor que hi ha en la posició a0 per l’immediat 0 en a3

a3 = 3

add a2,a2,a3

suma el valor de a3 mes el valor de a2 i guarda el resultat en a2

a2 = 3

addi a0,a0,4

suma l’immediat 4 a la posició on apunta a0, per tant de xx pasara a apuntar a yy

a0 = 0x10000004 (conte un 5)

addi a1, a1, 1

suma l’immediat 1 a a1 i guarda el resultat en a1

a1 = 1

j loop

fa un salt al loop per tornar a repetir les instruccions que hi ha dins

addi a7,a1,-4

suma el valor de a1 mes l’immediat -4 i guarda el resultat en a7

a7 = -3

beqz a7,final

si a7 es igual a 0 el programa saltarà a l’etiqueta final

a7 = -3 per tant no farà el salt

lw a3,0(a0)

carrega el valor que hi ha en la posició a0 per l’immediat 0 en a3

a3 = 5

add a2,a2,a3

suma el valor de a3 mes el valor de a2 i guarda el resultat en a2

a2 = 8

addi a0,a0,4

suma l’immediat 4 a la posició on apunta a0, per tant de yy pasara a apuntar a oo

a0 = 0x10000008 (conte un 2)

addi a1, a1, 1

suma l’immediat 1 a a1 i guarda el resultat en a1

a1 = 2

j loop

fa un salt al loop per tornar a repetir les instruccions que hi ha dins

addi a7,a1,-4

suma el valor de a1 mes l’immediat -4 i guarda el resultat en a7

a7 = -2

beqz a7,final

si a7 es igual a 0 el programa saltarà a l’etiqueta final

a7 = -2 per tant no farà el salt

lw a3,0(a0)

carrega el valor que hi ha en la posició a0 per l’immediat 0 en a3

a3 = 2

add a2,a2,a3

suma el valor de a3 mes el valor de a2 i guarda el resultat en a2

a2 = 10

addi a0,a0,4

suma l’immediat 4 a la posició on apunta a0, per tant de oo pasara a apuntar a zz

a0 = 0x1000000c (conte un 8)

addi a1, a1, 1

suma l’immediat 1 a a1 i guarda el resultat en a1

a1 = 3

j loop

fa un salt al loop per tornar a repetir les instruccions que hi ha dins

addi a7,a1,-4

suma el valor de a1 mes l’immediat -4 i guarda el resultat en a7

a7 = -1

beqz a7,final

si a7 es igual a 0 el programa saltarà a l’etiqueta final

a7 = -1 per tant no farà el salt

lw a3,0(a0)

carrega el valor que hi ha en la posició a0 per l’immediat 0 en a3

a3 = 8

add a2,a2,a3

suma el valor de a3 mes el valor de a2 i guarda el resultat en a2

a2 = 18

addi a0,a0,4

suma l’immediat 4 a la posició on apunta a0, per tant de zz pasara a apuntar a la seguent posicio de zz

a0 = 0x10000010 (conte un 0)

addi a1, a1, 1

suma l’immediat 1 a a1 i guarda el resultat en a1

a1 = 4

j loop

fa un salt al loop per tornar a repetir les instruccions que hi ha dins

addi a7,a1,-4

suma el valor de a1 mes l’immediat -4 i guarda el resultat en a7

a7 = 0

beqz a7,final

si a7 es igual a 0 el programa saltarà a l’etiqueta final

a7 = 0 per tant farà el salt

sw a2,4(a0)

guarda el contingut de a2 en a0 per l’immediat 4

a0 = 0x10000014 (conte el valor de a2 que es 18)

-4.

**Treball a fer a casa**

**h)**

RSCP: El programa triga 20 cicles en executar-se. Perquè cada instrucció es fa en un cicle i al main es fan 8 cicles, 2 per cada lw, lb i sw, al loop es fan 10 cicles, i finalment a l’últim lw es fan 2 cicles més, per tant 20 cicles en total.

R5SP: El programa triga 33 cicles en executar-se. Perque les instruccions es queden embussades

en el loop ja que les 5 fases no es reparteixen correctament les tres instruccions que hi ha dins del loop.

**i)**

RSCP:

**Gráfico, Gráfico de dispersión

Descripción generada automáticamente**

R5SP:

**Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza baja**

**j)**

RSCP:

El màxim nombre d’instruccions que s’han executat simultàniament es 1 ja que el RSCP executa les instruccions de una en una de manera seqüencial fins acabar el programa.

R5SP:

El màxim nombre d’instruccions que s’han executat simultàniament son 5 ja que el R5SP executa 5 instruccions a la vegada una darrere l’altre.

**Exercicis addicionals**

Teòrico-Pràctica 1:

**El mon binari**

**1. A partir del següents nombres binaris: A = 1001 0011 i B = 0011 1100.**

**Feu les següents operacions:**

A·B = 010 0010 0111 0100 Es fa la multiplicació aritmètica

A AND B = 0001 0000 Es fa la AND de A i B, quan un dels 2 es 0 el resultat es 0, quan els 2 son 1 el resultat es 1

A+B = 1100 1111 Es fa sa suma aritmètica

A OR B = 1011 1111 Es fa la OR de A i B, quan un dels 2 es 1 el resultat es 1 quan els 2 son 0 el resultat es 0

Comenteu els resultats

**Coma flotant**

**2. Transforma en representació de coma flotant el següent valor: 3,14**

3.14 = 0 10000000 10010001111010111000010

**2.1 Un cop trobat el valor, feu el procés invers per recuperar el valor**

1r. Mirem el bit de signe, com que es un 0 el numero serà positiu

2n. Els 8 bits següent corresponen a l’exponent 10000000 = 128 => 128 -127 = 1

3r. Mirem la matisa. m = 2-1 + 2-4 + 2-8 + 2-9 + 2-10 + 2-11 + 2-13 + 2-15 + 2-16 + 2-17 + 2-22 = 0.57

Per tant x = -10 · (1,57)·21 = 3.14

**Màscares**

**3. Feu un programa que ens indiqui si un determinat valor obtingut per una operació prèvia és positiu o negatiu fent servir màscares i operacions lògiques.**

MASCARA = 80h

operacio = XXh

if (operacio && MASCARA == 0):

print(‘valor és positiu’)

else:

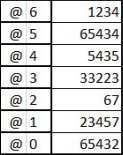
print (‘valor és negatiu’)

**4. Troba la màscara que has de fer servir per determinar que un número és negatiu. Considerant que el valor està donat en Ca2, passa-ho al valor positiu que correspondria (ex. -7 => 7). Troba vàries solucions al problema.**

La mascara que s’ha de fer servir per determinar que un número és negatiu és mascara = 80h = 10000000.

Per pesarà el valor en negatiu en Ca2 a positiu s’inverteixen els 1s i 0s i es suma 1 al resultar de fer la inversió.

**Punters**

**5. La instrucció MOV A, M guarda en el registre A el contingut que hi ha a la posició de memòria M. Indiqueu el contingut del registre A en funció dels possibles valors de M**

Per M = @6, A = 1234

Per M = @5, A = 65434

Per M = @4, A = 5435

Per M = @3, A = 33223

Per M = @2, A = 67

Per M = @1, A = 23457

Per M = @0, A = 65432

**Conclusions**

Amb aquesta practica he anat entenent les diferents instruccions d’accés a memòria, l’espai que ocupa una word a mes dels loops amb les condicions de salt i les operacions instruccions aritmètic – lògiques.

M’ha costat entendre com funciona tot el tema de la memòria i d’on surten els valors que es donen al PC.

Com que no hem vist molt el R5SP no acabo d’entendre el seu funcionament i com es complementen les cinc instruccions que s’executen simultàniament.

Nacho Rivera

Grup B