Pràctica 5
Estructures de control

5.1 Objectius

- Entendre la necessitat que hi haja instruccions per a trencar l'execució seqüencial dels programes.
- Conèixer la traducció al llenguatge d'assemblador de les estructures de control bàsiques de la programació estructurada.

5.2 Material

Simulador MARS i un codi font de partida.

5.3 Desenvolupament de la pràctica

5.3.1 Introducció

En un programa MIPS l'ordre d'execució de les instruccions és seqüencial, el registre *Comptador de Programa* (PC) conté l'adreça on es troba la següent instrucció a executar. Aquest valor s'incrementa automàticament amb 4 després de cada cicle d'instrucció com heu pogut comprovar en les pràctiques precedents.

No obstant això, hi ha vegades que pot interessar saltar i executar instruccions que no es troben en la seqüencia normal del programa. Heu vist ja que el MIPS proporciona instruccions per a poder-ho fer com la *jal* o la *j*, són el que s'anomenen *bots incondicionals*, és a dir, instruccions que sempre que s'executen realitzaran un bot que modifica el contingut del PC. A banda d'aquestes instruccions, el MIPS ens en proporciona d'altres que ens permeten trencar la seqüencia normal del programa depenent de si es compleix o no una certa condició, són les anomenades *instruccions de bot condicionals*. Combinant els dos tipus d'instruccions de bot (condicional i incondicional) podem programar estructures de control complexes, cosa que permet als compiladors

traduir sentències de la programació en alt nivell com if-then-else, while, for-loop.

L'estructura if-then-else és probablement la més utilitzada pels programadors. Vegem un exemple de la seua traducció a l'assemblador. Suposeu aquest fragment de codi en C:

```
if (a != b)
     f = g + h;
```

Es vol executar la suma si *a* és diferent de *b*, o el que és el mateix, si *a* es igual a *b* no es vol fer la suma. Aquesta sentència es pot escriure en assemblador del MIPS utilitzant la instrucció *beq* (*branch if equal*) de la manera següent:

```
        ##################################
        #                                #
        #     Codi exemple               #
        #     if-then-else               #
        #                                #
        ##################################


        beq $t1, $t2, Final     # si a=b salta a Final

        add $t3, $t4, $t5       # f = g + h

Final:  # Altres instruccions
```

La instrucció *beq* és un exemple d'instrucció de bot condicional. Perquè es puga fer el bot s'ha de satisfer la condició d'igualtat entre els registres indicats; altrament el programa continua executant-se en l'ordre normal. En l'exemple, si no se satisfà la condició ($t1=$t2) s'executa la següent instrucció en seqüencia que és l'*add*.

5.3.2 Com fer ús de les instruccions de bot condicionals

A continuació estudiarem com traduir a l'assemblador condicions lògiques simples utilitzades en la programació en alt nivell.

Anem a fer un programa que llija un enter del teclat i escriga en la consola el seu valor absolut. Per a obtenir el valor absolut s'utilitzarà una estructura de control del tipus if-then-else. L'estructura que tindrà el programa serà

```
Llegir el valor (direm A)
Si (A ≥ 0) anar a etiq
Fer A = –A
```

```
etiq:   Imprimir A
```

Per a la condició $A \geq 0$ utilitzarem la instrucció de bot condicional *bgez* (*branch if greater than or equal to zero*). El programa quedaria:

```
          ##################################
          #                                #
          #     Codi de l'activitat 1      #
          #     Càlcul del valor absolut   #
          #     d'un enter                 #
          #                                #
          ##################################


          .text

          li $a0, '>'    #Es demana introduir un enter
          li $v0,11
          syscall
          li $v0,5
          syscall              #Llig l'enter A

          bgez $v0, else       # Si (A ≥ 0) salta a else
          sub $a0, $zero, $v0  # En $a0 el negatiu de A
          j exit               # Acaba part if-then

else:     move $a0, $v0        # En $a0 el valor A

exit:     li $v0, 1            #Imprimim el valor en $a0
          syscall
          li $v0, 10           #Acaba el programa
          syscall
```

5.3.2.1 Activitat 1

- Escriviu, assembleu i comproveu el funcionament del codi *Càlcul del valor absolut d'un enter*.

- Observeu en la finestra *registers* com varia el contingut del registre PC.

Qüestió 1

- La instrucció *bltz* (*branch if less than zero*) salta si menor que zero i és similar a *bgez*, ja que també compara un registre amb 0 però sent ara contraria la condició de bot. Canvieu la instrucció *bgez* per *bltz* en el programa de l'activitat 1. Quines modificacions hauríeu de fer en el codi?

- Comproveu que el nou programa amb *bltz* opera correctament.

En el repertori d'instruccions del MIPS hi ha sis condicions per als bots condicionals amb els quals es poden fer tres parells de condicions

contràries (= i ≠, > i ≤, < i ≥). Amb les dues primeres condicions(= i ≠) s'han d'especificar dos registres (per exemple: `beq rs, rt, etiqueta`), són les instruccions *beq* i *bne*. Per a la resta de condicions sols s'ha d'especificar un registre en la instrucció perquè la comparació es fa amb el registre $zero que sempre conté un 0 i no cal indicar-lo (per exemple: `bltz rs, etiqueta`), són les instruccions *bgtz*, *blez*, *bltz*, *bgez*.

Què ocorre si volem fer un bot amb una condició entre dos registres diferent de la igualtat o desigualtat? El MIPS no ens proporciona les instruccions de bot que ens permeten fer-ho. El que sí que ens proporciona són instruccions de comparació que podem utilitzar combinant-les amb les instruccions de bot esmentades per a obtenir els nostres propòsits. Per exemple, la instrucció *slt* (*set on less than*) té la forma següent: `slt rd,rs,rt`. Aquesta instrucció compara dos registres (r*s* i r*t*) i posa un 1 en el registre r*d* si r*s* es menor que *rt*, en cas contrari situa en r*d* un 0.

Imagineu que volem traduir a l'assemblador la sentència següent en pseudocodi:

```
if $s2 ≥ $s1 then
        $s3 = 0
```

Ho farem utilitzant les instruccions *slt* i *beq*:

```
          ###################################
          #                                 #
          #     Codi exemple                #
          #     if-then-else   amb slt      #
          #                                 #
          ###################################


       slt $t0,$s1,$s2         #$t0=1 si $s1<$s2
       beq $t0,$zero,salta     #Salta si falla condició
       add $s3,$zero,$zero     $s3=0

salta:    # Resta d'instruccions
```

Qüestió 2

- Escriviu el programa que llig dos enters del teclat i escriu en la consola el més gran. El programa ha de tenir l'estructura següent:

```
          Llegir el primer valor (direm A)
          Llegir el segon valor (direm B)
          Si (A<B) anar a eti
             Imprimir A
             Anar a acabar:
 eti:     Imprimir B
```

```
acabar:
```

- Assembla i prova el programa amb diferents valors d'A i B

5.3.3 Codificació en llenguatge de màquina

Les instruccions de bot condicional es codifiquen en binari seguint el format tipus I per a emmagatzemar-les en la memòria. A diferència de les de bot incondicional, no assenyalen l'adreça a la que es saltarà sinó que indiquen el desplaçament en bytes fins on es vol arribar; aquest desplaçament s'afegirà al valor que hi haja emmagatzemat en el PC per a aconseguir l'adreça correcta.

Qüestió 3

- Observeu el programa assemblat que acabeu d'escriure corresponent a la qüestió 2 i ompli la plantilla de la figura 1 amb els camps en binari de la instrucció *beq* segons el format tipus I; escriviu primer la instrucció que codificareu.

Instrucció: .....

Codi Operació (6 bits)	Rs (5 bits)	Rt (5 bits)	Desplaçament

Figura 1. Plantilla a omplir de la qüestió 3

Les instruccions de comparació són instruccions aritmètiques que impliquen tres registres; segueixen, per tant, el format tipus R.

Qüestió 4

- Observeu de nou el programa assemblat que acabeu d'escriure corresponent a la qüestió 2 i ompliu la plantilla de la figura 2 amb els diferents camps en binari de la instrucció *slt* segons el format R. Escriu primer la instrucció a codificar.

Instrucció:.....

Codi op (6 bits)	Rs (5 bits)	Rt (5 bits)	Rd (5 bits)	Shamt(5 bits)	Funció (6 bits)

Figura 2. Plantilla a omplir de la qüestió 4

5.3.4 Ajudes a la programació, pseudoinstruccions

Combinant instruccions de comparació amb instruccions de bot condicional, el MIPS ens proporciona una sèrie de pseudointruccions de

bot condicional amb les quals ens podem recolzar per fer programes més llegibles. El fet que hi haja poques instruccions bàsiques de bot es deu a la decisió presa pels dissenyadors del MIPS de no utilitzar codis d'operació del format tipus I innecessàriament sinó reservar-los per a instruccions més importants.

Així, per exemple, el MIPS disposa de la pseudoinstrucció `bgt Rs, Rt ,etiqueta`, la qual es tradueix per la següent parella d'instruccions.

```
slt $at,Rt,Rs
bne $at,$zero,Etiqueta
```

S'ha de tenir en compte que, quan fa la substitució d'aquests tipus de pseudoinstruccions, l'assemblador utilitza el registre *$at* per a guardar el resultat de la comparació. Aneu amb compte i no utilitzeu el registre *$at* quan programeu perquè l'assemblador podria destruir el valor que heu ficat en aquest registre.

En la tabla 1 s'han agrupat les instruccions de bot condicional juntament amb algunes pseudoinstuccions.

Instrucció	Exemple	Significat	Comentaris
Branch if equal	beq Rs, Rt, etiqueta	Si (Rs = Rt)⇨ PC ← etiqueta	Salta a etiqueta si Rs és igual a Rt
Branch if Greater Than or Equal to Zero:	bgez Rs, etiqueta	Si (Rs ≥ 0) ⇨ PC ← etiqueta	Salta a etiqueta si Rs és major o igual que 0.
Branch if Greater Than Zero:	bgtz Rs , etiqueta	Si (Rs >0) ⇨ PC ← etiqueta	Salta a etiqueta si Rs és major que 0.
Branch if Less Than or Equal to Zero:	blez Rs ,etiqueta	Si (Rs ≤0) ⇨ PC ← etiqueta	Salta a etiqueta si Rs és menor o igual que 0.
Branch if Less Than Zero:	bltz Rs , etiqueta	Si (Rs <0) ⇨ PC ← etiqueta	Salta si Rs menor que 0.
Branch if Not Equal:	bne Rs, Rt , etiqueta	Si (Rs ≠ Rt) ⇨ PC ← etiqueta	Pseudoinstrucció. Salta a etiqueta si Rs no és igual a Rt.
Branch if Greater Than or Equal	bge Rs, Rt , etiqueta	Si (Rs ≥ Rt) ⇨ PC ← etiqueta	Pseudoinstrucció. Salta a etiqueta si Rs és major o igual que Rt.
Branch if Greater Than:	bgt Rs, Rt , etiqueta	Si (Rs > Rt) ⇨ PC ← etiqueta	Pseudoinstrucció. Salta a etiqueta si Rs és major que Rt.
Branch if Less Than or Equal:	ble Rs , Rt , etiqueta	Si (Rs ≤ Rt) ⇨ PC ← etiqueta	Pseudoinstrucció. Salta a etiqueta si Rs és major o igual que Rt .

Branch if Less Than:	blt Rs , Rt , etiqueta	Si (Rs < Rt) ⇨ PC ← etiqueta	Pseudoinstrucció. Salta a etiqueta si Rs és menor que Rt.

Taula 1. Instruccions i pseudoinstruccions de bot condicional

En la taula 2 es mostren les instruccions de comparació amb algunes pseudoinstruccions:

Instrucció	Exemple	Significat	Comentaris
Set on Less Than	slt Rd, Rs, Rt	Rd← 1 si Rs < Rt si no Rd← 0	Compara menor que i posa Rd a 1 si es compleix; complement a 2
Set on Less Than Immediate:	slti Rd, Rs ,k	Rd← 1 si Rs < k si no Rd← 0	Compara menor que i posa Rd a 1 si es compleix; complement a 2
Set on Less Than unnsigned	sltu Rd, Rs, Rt	Rd← 1 si Rs < Rt si no Rd← 0	Compara menor que i posa Rd a 1 si es compleix; considera valors sense signe
Set if Equal:	seq Rd, Rs, Rt	Rd← 1 si Rs = Rt si no Rd← 0	Pseudoinstrucció. Compara si igual i posa Rd a 1 si es compleix; complement a 2
Set if Greater Than or Equal:	sge Rd, Rs, Rt	Rd← 1 si Rs ≥ Rt si no Rd← 0	Pseudoinstrucció. Compara si major o igual, posa Rd a 1 si es compleix; complement a 2 .
Set if Greater Than:	sgt Rd, Rs, Rt	Rd← 1 si Rs > Rt si no Rd← 0	Pseudoinstrucció. Compara major que i posa Rd a 1 si es compleix; complement a 2 .
Set if Less Than or Equal:	sle Rd, Rs, Rt	Rd← 1 si Rs ≤ Rt si no Rd← 0	Pseudoinstrucció. Compara menor o igual i posa Rd a 1 si es compleix; complement a 2 .
Set if Not Equal:	sne Rd, Rs, Rt	Rd← 1 si Rs ≠ Rt si no Rd← 0	Pseudoinstrucció. Compara si no igual que i posa Rd a 1 si es compleix; complement a 2 .

Taula 2. Instruccions i pseudoinstruccions de comparació

Qüestió 5

- Escriviu un codi exemple per a esbrinar com tradueix l'assemblador les pseudoinstruccions següents: *sgt*, *sge*, *ble*.

Qüestió 6

- Penseu en un cas en el qual *slt* i *sltu* donen resultats diferents per als mateixos continguts dels registres. Comproveu-ho.

5.3.5 Bucles

Hi ha dues estructures de bucle simples en la majoria dels llenguatges de programació; són les conegudes com a *while-loop* i *for-loop*.

Un bucle *while* es caracteritza per tenir una sentència que controla si el bucle s'executarà o no; vegem-ne un exemple. Suposeu aquesta sentència de pseudocodi:

```
while (s1 ≤ s2)
        s1 = s1 + s5;
```

volem eixir del bucle si falla la condició, és a dir, si *s1>s2*, o escrit d'una altra manera, si *s2<s1*. Els passos que s'hauran de seguir per traduir-lo a l'assemblador seran sempre els mateixos:

1. La sentència que comprova si s'entra en el bucle es posa a l'inici.

2. Hi haurà una etiqueta per a començar el bucle; d'aquesta manera, l'última instrucció del bucle serà un bot incondicional a ella per començar de nou.

3. Hi haurà una segona etiqueta per acabar el bucle.

4. Se saltarà a ella quan la condició de comprovació de no entrada al bucle es complisca.

El codi escrit en assemblador del MIPS quedaria de la manera següent:

```
        ##################################
        #                                #
        #     Codi exemple               #
        #     Bucle WHILE                #
        #                                #
        ##################################


BucleWhile: slt $t0, $s2, $s1       #Comprovem si s2<s1
            bne $t0, $zero, EixirBucle #Si s2<s1 eixim
            add $s1, $s1, $s5
            j BucleWhile   # Torna a l'inici del bucle

EixirBucle:  #Eixida del bucle
```

Un bucle del tipus *for-while* es aquell que es considera que s'executarà un nombre determinat de vegades. La forma normal d'expressar-lo és establint un valor d'inici per al comptador i incrementar-lo en cada operació del bucle. La condició de finalització serà quan el comptador arriba a un valor predeterminat.

Un exemple és el següent codi que suma els valors d'1 a 10:

```
n=11          # n-1, condició de finalització
total = 0;    # Inici del valor total de la suma
for (i = 1; i < n; i++)
{
        total = total + i
}
```

El bucle conté 3 parts, la primera es la inicialització de l'índex que s'ha de fer abans del bucle (*i=1*). Després trobem la condició de continuar dins del bucle (*i<n*). I la tercera és la condició que ens diu com incrementar el comptador (*i++*). Quan traduïm l'estructura a l'assemblador hem d'incloure les tres parts.

```
          ####################################
          #                                  #
          #      Exemple Bucle FOR-WHILE     #
          #                                  #
          ####################################

          li $s0, 1         #Iniciem índex
          li $s1, 11        #Condició de finalització
          li $s2,0          #Inicialització del comptador

inici_for:sle $t1, $s0, $s1    #Condició continuar dins
          beqz $t1, final_for  #si index=11 s'acaba bucle
          add $s2, $s2, $s0    #Cos del bucle
          addi $s0, $s0, 1     #increment de l'índex
          j inici_for          #Bot a l'inici del bucle

final_for:
```

Qüestió 7

- En exemples com l'anterior, en què sabem que almenys el bucle s'executa una vegada, és més eficient posar la condició d'eixida del bucle al final. Reescriviu el codi anterior amb la condició d'eixida al final del bucle (es tracta d'un bucle del tipus *do-while*).

Qüestió 8

- Afegiu al codi de la qüestió 7 la possibilitat de llegir el valor *n* per teclat i escriure el resultat de la suma en la consola. Feu que hi haja un bucle infinit que llija pel teclat llevat que s'escriga el valor 0; en aquest cas s'acaba el programa.

Qüestió 9

- Feu el codi que llija dos enters de la consola i n'escriga la suma i torne a començar si el resultat de la suma és diferent de 0. El pseudocodi seria:

```
(bucle do-while)
seguir: Llegir el primer valor (direm A)
        Llegir el segon valor (direm B)
        Imprimir A+B
        Si (A+B) == 0 anar a acabar
        anar a seguir
acabar:
```

Qüestió 10

- Feu el codi que llig de teclat dos valors positius A i B en els quals A<B. El programa ha d'escriure per consola els valors compresos entre ells inclosos ella mateixa. Es a dir, si A=3 i B=6, escriu en la consola 3 4 5 6 (podeu escriure, per exemple, un salt de línia després de cadascun dels valors a mostrar).

5.4 Resum

- Les instruccions de bot ens permeten trencar l'execució seqüencial dels programes.
- Les estructures de control d'alt nivell es poden traduir de manera relativament senzilla a l'assemblador combinant les instrucció de bot condicional e incondicional.
- Les pseudoinstruccions de bot combinen instruccions aritmètiques amb instruccions de bot condicional.