Escola d'Enginyeria, Enginyeria Telecomunicació

ACiP SOLUCIÓ PROBLEMES TEMA 1

Problema 1.

L'objectiu d'aquest problema consisteix en traduir fragments de codi en C a llenguatge assemblador.
En els codis escrits en C no declarem les variables que es fan servir. Suposem que aquestes variables ja les tenim declarades i en tots els casos són nombres **enters sense signe de 32 bits** codificats en complement a dos. Tampoc heu de reservar espai en memòria ni inicialitzar les variables en la memòria de l'assemblador. Per referir-nos a l'adreça de memòria d'una variable que surt en el codi en C utilitzem el mateix nom de la variable en C. Per exemple, per traduir la següent sentència en C,

A = A + B;

podem fer servir el codi següent:

```
LLA R1, A
LW R2, 0(R1)
LLA R3, B
LW R3, 0(R3)
ADD R3, R2, R3
SW R3, 0(R1)
```

Traduïu a llenguatge assemblador cadascun dels següents fragments de codi.

b)

```
while (A<B)
{
        C = C + C;
        A = A + 1;
}
```

Opció 1:

```
        LLA  R1, A        ; R1 ← &A
        LW   R2, 0(R1)    ; R2 ← [A]
        LLA  R3, B        ; R3 ← &B
        LW   R3, 0(R3)    ; R3 ← [B]
      ;;While (A < B) {
while:
        BGEU R2,R3,  endwhile   ; Salt al final del bucle while si R2 >= R3
      ;;      C = C + C
        LLA  R4, C        ; R4 ← &C
        LW   R5, 0(R4)    ; R5 ← [C]
        ADD R5, R5,R5     ; R5 ← [C]+[C]
        SW   R5, 0(R4)    ; [C] <- R5 , [C] <- [C]+[C]
      ;;      A = A + 1
        LLA  R1, A        ; R1 ← &A
        LW   R2, 0(R1)    ; R2 ← [A]
        ADDI R2, R2,1     ; R2 ← [A]+1
```

```
        SW  R2, 0(R1)    ; [A] □ R2 , [A] □ [A]+1
        J while           ; Tornem a la condició del while
      ;;}
endwhile:
```

Opció 2:

```
        LLA  R1, A       ; R1 ← &A
        LW   R2, 0(R1)   ; R2 ← [A]
        LLA  R3, B       ; R3 ← &B
        LW   R4, 0(R3)   ; R4 ← [B]
        LLA  R5, C       ; R5 ← &C
        LW   R6, 0(R5)   ; R6 ← [C]

;;While (A < B) {
while:
        BGEU R2,R4,  endwhile   ; Salt al final del bucle while si R2 >= R4
      ;;     C = C + C
        ADD R6, R6,R6           ; R6 emmagatzema els canvis que sofrirà la
                                ;variable C
      ;;     A = A + 1
        ADDI R2, R2,1           ; R2 emmagatzema els canvis que sofrirà la
                                ; variable A
        J while                 ; Tornem a la condició del while
      ;;}
      ; Fora del bucle guardem els últims valors de A i C, que son els
      ; que realment ens interessen (lectures/escriptures en memòria
      ; força més lentes que les operacions amb registres)
endwhile:
        SW R2, 0(R1)    ;Guardem el contingut de A
        SW R6, 0(R5)    ;Guardem el contingut de C
```

Problema 2.

Escriure un fragment de codi en llenguatge assemblador que realitzi la funcionalitat que es descriu en cadascun dels següents apartats:

a) Emmagatzemi en R1 un 1 si el contingut de R2 és més gran o igual que el de R3 i a més el de R4 és més petit que el de R5. Si no passa tot l'anterior, emmagatzemi un 0. Operem amb **nombres amb signe.**

```
;;if (R2 >= R3) && (R4 < R5)

BLT R2, R3, else     ; Si R2 < R3, AND es falsa, anem al else
BGE R4, R5,else      ; Si R4 >= R5, AND es  falsa, anem al else
```

```
        ;;{
        ;;      R1=1;
        LI R1, 1
        J out_if
        ;; }
 else:
        ;; else {
        ;;      R1 =0;
        LI R1, 0
        ;; }
out_if:
```

b) Emmagatzemi en R1 un 1 si es compleix una i només una de les dues condicions següents, Operem amb **nombres sense signe** :

- El contingut de R2 es més gran o igual que el de R3
- El contingut de R4 es més petit que el de R5.

En qualsevol altre cas emmagatzemeu en R1 un 0.

Opció 1: (Codi C amb AND's i OR, mes complex)

```
        ;;if ((R2 >= R3) && (R4 >= R5) ||  (R2 < R3) && (R4 < R5))

        BLTU R2, R3,or_1   ; Si R2 < R3, 1ª AND ja es falsa, anem a or_1
        BLTU R4, R5, else  ; Sinó, R2>=R3. Passo a comparar R4 i R5
                           ; Si R4 < R5, 1ª AND és falsa, anem a else i
                           ; la 2ª AND també és falsa, malgrat que la 2ª
                           ; condició (R4 < R5) d'aquesta 2ª AND es certa
                           ; la seva 1ª condició, que es R2 < R3, no ho és.
                           ; (venim del cas que  R2>=R3)
        J ok_if            ; Sinó, R4>=R5, es compleix la 1ª AND, OR és
                           ; certa.
 or_1:
        BGEU R4, R5,else   ; Venim de R2<R3, Si R4 >= R5, 2ª AND és falsa,
                           ; OR és falsa, anem a else
ok_if:                     ; Sinó, R4<R5,   2ª AND és certa, OR és  certa.

        ;;{
        ;;      R1=1;
        LI R1, 1
        J out_if
        ;;}
 else:
        ;; else {
        ;;      R1 =0;
        LI R1, 0
```

```
            ;;}
 out_if:
```

Opció 2: (Codi més senzill i ràpid)

```
            ;;R6=1;
            LI R6, 1          ; R6 ← 1, Suposem que es compleix R2>=R3
            ;;if (R2<R3) {
            BGEU R2, R3, continue ; Si R2>=R3, és compleix una condició
                ;;R6=0;
            LI R6,0     ; Sinó R6 ← 0, indicant que R2<R3.
            ;;}
            ;;R7=1;
Continue1:
            LIR7, 1       ; R7 ← 1, Suposem que se compleix R4<R5
            ;;if (R4>=R5) {
            BLTU R4, R5, continue2   ; Si R4<R5, és compleix una condició
                ;; R7=0;
            LI R7, 0       ; Sinó R7 ← 0, Indicant que R4>=R5.
            ;;}
Continue2:
            ;;R1=R6 ^ R7;
            XOR R6, R6, R7 ; Casos:
                           ; 1) Les 2 condicions es compleixen : R6 i R7 = 1
                           ;     1 XOR 1 = 0, Correcte ja que només una pot ser
                           ;     R6= 1 → (R2>=R3), R7=1 → (R4<R5), cap AND
                           ;     es certa sota aquestes condicions.
                           ; 2) Una de les 2 condicions compleix: R6 o R7 són 1
                           ;     0 XOR 1 o 1 XOR 0 es 1, Correcte.
                           ;     R6= 1 → (R2>=R3), R7=0 → (R4>=R5), la 1ª
                           ;     AND    es certa, per tant la OR es certa.
                           ; 3) Cap condició compleix: R6 i R7 = 0
                           ;     0 XOR 0 = 0, Correcte ja que només pot ser una
                           ;     i en aquest cas no compleix cap
                           ;    R6= 0 → (R2<R3), R7=0 → (R4>=R5), cap AND
                           ;     es certa sota aquestes condicions.
            MV R1, R6
```