Tema 3. Enunciats dels problemes

Joan Manuel Parcerisa

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Problema 3.3

3.3. Escriu les seqüències de codi necessàries per escriure en $t1 el resultat de les següents condicions lògiques, sense fer servir instruccions de salt. El valor final de $t1 ha de ser 0 si no s'acompleix la condició, o diferent de zero altrament.

a) L'enter representat per $t1 és negatiu

b) L'enter representat per $t1 és múltiple de 4

c) El natural representat per $t1 és tal que el seu quadrat no es pot escriure amb 32 bit

d) El natural representat per $t1 és tal que la suma amb si mateix no es pot escriure amb 32 bits

e) Els bits 1, 3 i 5 de $t1 valen zero

Problema 3.4

3.4.

```
sllv rd, rt, rs      # rd=rt<<rs (anàleg a sll)
srlv rd, rt, rs      # rd=rt>>rs (anàleg a srl)
srav rd, rt, rs      # rd=rt>>rs (anàleg a sra)
```

Fent servir alguna d'aquestes instruccions, programa una seqüència de no més de 6 instruccions (es pot fer amb 4) que serveixi per posar a 0 el bit *i*-èssim del registre $t1, deixant la resta de bits intactes, i suposant que *i* està guardat en $t2 (pots usar d'altres registres per guardar resultats temporals).

Problema 3.5

3.5. Fes un programa que realitzi els següents desplaçaments sobre un número de 64 bits emmagatzemat als registres \$t2 (32 bits de més pes) i \$t1 (32 bits de menys pes). La clau de l'exercici és calcular en un registre a part el(s) bit(s) que es desplacen d'un registre a l'altre, alineant-lo(s) a dreta o esquerra segons convingui:

a) Shift a l'esquerra 1 posició

d) Fent servir les instruccions de shift variable explicades al problema anterior, fer un shift lògic a l'esquerra, n posicions (n<32), on n ocupa el registre \$t3.

Problema 3.16

3.16. Tradueix a llenguatge assemblador MIPS el següent programa en C, suposant que les variables *a*, *b*, *c* ja han estat inicialitzades i ocupen els registres *$t1*, *$t2* i *$t3* respectivament:

```
main() {
      int a, b, c, i;          /* i s'emmagatzema en $t0  */
      ...                      /* inicialització de les variables */
      if (a>0)
      {
               i = b;
               if (((i-4) < a) && (i>0))
                        b = b+c;
               else if (c>0)
                        b = b-c;
      }
      else
               a = a+b;
}
```

Problema 3.22

3.22. Tradueix a assemblador MIPS la següent funció recursiva:

```
int pellnumber(int n) {
     if (n < 2)
              return n;
     else
              return 2*pellnumber(n-1) + pellnumber(n-2);
}
```

Exercici subrutines amb bucle

- Tradueix a MIPS la següent funció f, escrita en C

```
int g(char *a, int b);
char f(int n) {
      char v[20];
      int i, x=0;
      for (i=0; i<n; i++){
            x = g(v, x);
      }
      return v[x];
}
```

Problema 3.28

3.28. Donades les següents declaracions, tradueix a assemblador MIPS la funció *s1*:

```
int s2 (int c, long long *d, long long w[]);
int s1 (int x, long long v[], int *p)
{
     long long *k;
     k = &v[x];
     x = s2(*p, k, v);
     return *p + x;
}
```

Problema 3.33

3.33. Donades les següents declaracions, tradueix a assemblador MIPS la funció *subr1*.

```
int subr2(int a, char *b, char c, int d);

int subr1(char v[], int e2, int *var, char p)
{
     int valor;
     valor = subr2(*var, &v[e2], p, e2);
     return valor + e2;
}
```

Exercici subrutines

- Donat el següent codi en C
  - Dibuixa el bloc d'activació
  - Tradueix a MIPS la sentència: `y = x + g(w, &y, v[i]);`

```
char g(int *a, char *b, int c);
char f(int w[], char *p, int i)
{
        char x, y, z;
        int v[10];
        x = g(v, p, i);
        y = x + g(w, &y, v[i]);
        z = g(v, &y, i);
        return y + z;
}
```

Exercici operacions amb bits

- Una funció té les següents variables locals guardades en $s0, $s1, $s2:

```
int *a, b, c;
```

- Tradueix a assemblador MIPS la següent sentència en C:

```
*(a+1) = (b & 1) || (c >> 1);
```

```
        [          ] $t0, $s1, 1
        [          ] $t0, $zero, fi
        [          ] $t1, $s2, 1
        [          ] $t0, $zero, $t1
fi:     sw [                    ]
```

Exercici anàlisi de codi

- Escriu el valor final en hexadecimal del registre $t0

```
li   $t0, 0x0020A040
sw   $t0, 0($sp)
lb   $t0, 1($sp)
srl  $t0, $t0, 4
ori  $t0, $t0, 0x0099
```

Parcial 2018/19-Q1

Un programa està compost de dos mòduls que es compilen i assemblen separadament per generar sengles fitxers objecte. Per a generar l'executable cal enllaçar-los després amb el muntador. El codi en C dels dos mòduls és el següent:

MODUL 1: `int main() { f(V[X]); }`

MODUL 2: `void f(int i) { Y=i; }`

Les variables `V`, `X`, `Y` són globals. Hem traduït els dos fitxers a MIPS amb el següent resultat (hem afegit a l'esquerra els números de línia per facilitar les respostes posteriors):

MÒDUL 1

```
1         .data
2         .globl V
3   V:    .word 1, 2, 3, 4, 5
4   Y:    .word 0

5         .text
6         .globl main
7   main:addiu  $sp, $sp, -4
8         sw    $ra, 0($sp)

9         la    $t0, X
10        lw    $t1, 0($t0) # $t1 <- X
11        la    $t2, V
12        sll   $t3, $t1, 2
13        addu  $t4, $t2, $t3
14        lw    $a0, 0($t4) # $a0 <- V[X]
15        jal   f

16        lw    $ra, 0($sp)
17        addiu $sp, $sp, 4
18        jr    $ra
```

MÒDUL 2

```
1         .data
2         .globl  X
3   X:    .word   3

4         .text
5         .globl  f
6   f:    la      $t0, Y
7         sw      $a0, 0($t0) # Y <- $a0
8         jr      $ra
```

Parcial 2018/19-Q1

a) Quan hem intentat enllaçar els dos fitxers objecte generats per l'assemblador, l'enllaçador ha detectat un error. Com caldrà corregir el codi MIPS perquè no torni a fallar?

mòdul: codi corregit:

b) Contesta les següents 3 preguntes suposant que s'ha corregit l'error anterior i que conservem la numeració de línies original:

Pregunta	MÒDUL 1	MÒDUL 2
Quines etiquetes conté la Taula de Símbols Globals de cada fitxer objecte?		
Quines línies de codi en cada fitxer objecte (sols el número) contenen referències **no-resoltes** (referències creuades)?		
Quines línies de codi en cada fitxer objecte (sols el número) contenen **adreces absolutes** (i necessiten ser reubicades)?		