Edgar Aguadé Nadal.

EC Examen de Problemes

Exercici 1 (Ex. Parcial 2013-2014 Q1)

Tradueix a llenguatge assemblador MIPS la subrutina func1:

```
short func2(short a, char *b, short *c);

short func1 (int x, short *y) {
  char V1[7];
  short V2[7];
  short res;
  res=func2(*y,&V1[x],V2);
  if (x>0)
    res++;
  return res+(*y);
}
```

Exercici 2

Donades les següents declaracions en C (on N és una constant):

```
#define N 10

int A[N][N];

void func(int B[][N]) {
  int i, j; // a $t1 i $t2 respectivament.
    ... /* aquí va la sentència de cada apartat */
}
```

Tradueix a MIPS les següents sentències, suposant que pertanyen a la funció func:

a) `B[i][3] = 0;`

b) `B[i][j] = 0;`

c)
```
for (i=0; i<N; i++) //utilitza accés seqüencial
    B[3][i] = 0;
```

d)
```
for (i=0; i<N; i++) //utilitza accés seqüencial
     B[i][i] = 0;
```

e)
```
for (i=0; i<N; i++) //utilitza accés seqüencial
    B[i][N-1-i] = 0;
```

Exercici 3 (Ex. Final 2011-2012 Q2)

Considera el següent programa

```
int v[20],m[20][20];
main() {
    int i;
    for (i=19; i>=0; i--)
        v[i] = m[19-i][i];
}
```

Tradueix el programa principal a llenguatge assemblador MIPS. Només superaran aquesta pregunta aquelles solucions en què cada iteració del bucle tingui 7 o menys línies de codi.

Edgar Aguadé Nadal.

EC Examen de Problemes. 2013-2014 Q1.

Ex 1 8

BA8

0($sp)	V1	7 byte	$s0 ← x
	//////	1 byte	$s1 ← *y
8($sp)	V2	7*2 byte	$ra
	//////	2 byte	
24($sp)	$s0old	4 byte	
28($sp)	$s1old	4 byte	
32($sp)	$raold.	4 byte	

Total 8 36 bytes. 4|36 ⇒ cert.

```
addiu $sp, $sp, -36  # reserva B.A.
sw $s0, 24($sp)      # $s0 old
sw $s1, 28($sp)      # $s1 old
sw $ra, 32($sp)      # $ra old.
move $s0, $a0        # $s0 ← x
move $s1, $a1        # $s1 ← *y.

lh $a0, 0($s1)
addu $a1, $sp, $s0
addu $a2, $sp, 8     # 8($sp)
jal func2.

if:        $s0   $zero
   ble $zero, $s0, fi_if
      addiu $v0, $v0, 1
fi_if:
   lh $t0, 0($s1)
      addi $v0, $v0, $t0  # preparem el return.

   lw $s0, 24($sp)
   lw $s1, 28($sp)
   lw $ra, 32($sp)      # recuperem valors antics
   addiu $sp, $sp, 36
   jr $ra # return.
```

Ex 38 stride: m[19-(i-1)][i-1] = m + (19-i+1)*20 + i + 1

m[19-i][i] = m + (19-i)*20 + i

-20+1 = -19 elements (76 bytes)

```
main8
  la $t1, v
  la $t2, m
  addiu $t1, $t1, 76 # &v[19]
  addu $t2, $t2, 76  # &m[0][19]
  li $t0, 19 # i=19.
_for8
  blt $t0, $zero, end_for.
  lw $t7, 0($t2)
  sw $t7, 0($t1) # v[i] = m[19-i][i]
  addiu $t1, $t1, -4 # v[i+1]
  addiu $t2, $t2, 76 # m[19+(i+1)][i+1]
  addiu $t0, $t0, -1 # --i
  b_for
_end_for:
```

Edgar Aguadé Nadal.

Ex 28

a) B[i][3] = 0

```
addiu $t0, $a0, 12 # &B[0][3] (12 = 3*4)
li $t7, N*4 # N=10
mult $t1, $t7 # i*N*4.
mflo $t1
addu $t0, $t0, $t1 # B+i*N*4 + 3*4.
sw $zero, 0($t0) # B[i][3] = 0.
```

b) B[i][j] = 0;

```
li $t7, N*4 #N=10
mult $t1, $t7 #i*N*4 -> mflo $t1
sll $t2, $t2, 2 #j*4.
addu $a0, $a0, $t1 # B+i*N*4
addu $a0, $a0, $t2 # B+i*N*4 + j*4.
sw $zero, 0($a0)    # B[i][j] = 0.
```

c)

```
li $t0, 0 #i                    (per calc.)
li $t7, 3*N*4 # 120 (&B[3][0])
addu $a0, $a0, $t7 # &B[3][0]
_for8
slti $t1, $t0, 10 #i < N.
beq $t1, $zero, _fi_for:
 sw $zero, 0($a0)
addiu $a0, $a0, 4 #& B[3][i+1]
b for
_fi_for:
```

d) STRIDES B[i+1][i+1] = B + i*N + 1*N + i + 1
            B[i][i]   = B + i*N + i
} N+1 elements, 4N+4 bytes.

```
li $t7, 4*(N+1) #44.
li $t0, 0
_for8
slti $t1, $t0, 10 #i < N
beq $t1, $zero, end_for.
sw $zero, 0($a0) #B[i][i] = 0
addu $a0, $a0, $t7 #&B[i+1][i+1]
b _for8
_end_for8
```

e) Stride: B[i][N-1-i] = -(B + i*N + N - i - 1)
           B[i+1][N-1-i-1] = B + i*N + 1*N + N - i - 2   (i - 2)
} N-1 elements, 4(N-1) bytes.

```
li $t7, 4*(N-1) #36
addiu $a0, $a0, 36 #9*4 (&B[0][9])
li $t6, 0
_for8
slti $t1, $t0, 10 # i < 10
beq $t1, $zero, _end_for.
```

```
sw $zero, 0($a0)
addu $a0, $a0, $t7 #&B[i+1][(N-1 ... +1)
b _for
_end_for8
```