COMP2611: Computer Organization

MIPS branch and jump instructions

Exercises

Question 1: Write down the MIPS instructions for the following C++ codes, assuming each variable is stored in a different register (you name it). You can use some registers for storing temporary values.

```
if (d < 4) {
  if (d == 1)
     d = d + 4;
  else ++d;
}
```

One possible solution:

```
slti $t0, $s0, 4    #s0 stores d
beq $t0, $zero, IfEnd
addi $t1, $zero, 1
bne $s0, $t1, Else
addi $s0, $s0, 4
j IfEnd
Else: addi $s0, $s0, 1
IfEnd:
```

Question 2: Write down the MIPS instructions for the following C++ codes, assuming each variable is stored in a different register (you name it). You can use some registers for storing temporary values.

```
switch (d) {
  case 1: d = d + 4;
          break;
  case 4: d = d * 2;
          break;
  default: d--;
}
```

One possible solution:

```
slti $t0, $s0, 4    #s0 stores d
beq $t0, $zero, IfEnd
addi $t1, $zero, 1
bne $s0, $t1, Else
addi $s0, $s0, 4
j IfEnd
Else: addi $s0, $s0, 1
IfEnd:
```

Question 3: Write down the MIPS instructions for the following C++ codes, assuming the base address of the array `A` of int elements is stored in the register `$s1` and each variable is stored in a different register (you name it). You can use some registers for storing temporary values.

```
c = 10;
while (c >= 10 && c <= 20)
{
    if (c < 15)
     A[c - 4] = A[c + 3] - c;
    c++;
}
```

Exercises

Solution to question 3

```
#One Possible Solution
        addi $s0, $zero, 10     #$s0 stores c
        addi $t0, $zero, 10     #$t0 stores 10
        sll $t1, $s0, 1         #$t1 stores 20
Loop:
        blt $s0, $t0, Exit
        bgt $s0, $t1, Exit
        slti $t2, $s0,15
        beq $t2, $zero, L1
        addi $t3, $s0, 3
        sll $t3, $t3,2
        add $t3, $t3, $s1       #address of A[c+3]
        lw $t4, 0($t3)
        sub $t4, $t4, $s0       #$t4 has: A[c+3]-c
        sub $t5, $s0, 4
        sll $t5, $t5,2
        add $t5, $t5, $s1       #address of A[c-4]
        sw $t4, 0($t5)
#using sw $t4, -28($t3) to replace the above four instructions
L1:   addi $s0,$s0, 1
        j  Loop
Exit:
```

Question 4: Write down the MIPS instructions for the following C++ code, assume the base address of an `int` array `A` is stored in the register `$s1` and each variable is stored in a different register (you name it). You can use some registers for storing temporary values.

```
c = 0;
do {
      c = c + 2;
      A[c - 1] = A[c];
} while (c < 10);
```

One possible solution:

```
        addi $s0, $zero, 0        #$s0 stores c
Loop:   addi $s0, $s0, 2
        sll $t1, $s0, 2           #$t1 has $s0*2^2
        add $t1, $s1, $t1         #$t1 has the addr of
                                    A[c]
        lw $t2, 0($t1)
        sw $t2, -4($t1)
        slti $t0, $s0, 10
        bne $t0, $zero, Loop
```

Exercises

Question 5: Write down the MIPS instructions for the following C++ code, assume the base address of an int array `A` is stored in the register `$s1` and each variable is stored in a different register (you name it). You can use some registers for storing temporary values.

```
for (int c = 0; c <= 10; c += 2)
{
    A[c] = A[c + 3];
}
```

One possible solution:

```
addi $s0, $zero, 0   #$s0 stores c
Loop: slti $t0, $s0, 11
      beq $t0, $zero, LoopEnd
       sll $t0, $s0, 2
      add $t0, $s1, $t0
      lw $t1, 12($t0)
      sw $t1, 0($t0)
       addi $s0, $s0, 2
       j Loop
LoopEnd:
```

Exercises

Exercise 1: Write down MIPS instructions for the following C++ statements. Assume the variables `i, j, x,` and `y` are stored in the registers `$t0, $t1, $a1,` and `$a2`.

```
int i = 0;
 int j = -1;
 while ( i < 10) {
    if ((i & 0x0001) == 1)
        j+=i;
    i++;
 }
```

One possible solution:

```
        add $t0, $zero, $zero    #$t0 stores i
        addi $t1, $zero, -1      #$t1 stores j
Loop: slti $t2, $t0, 10
        beq $t2, $zero, End
        andi $t3, $t0, 1      #$t0 is odd, j+=i
        beq $t3, $zero, Notif
        add $t1, $t1, $t0     #j+=i
Notif:
        addi $t0 $t0, 1       #$t0 is even, i++
        j loop
End:
```

Exercise 2: Write down the MIPS instructions to find the Maximum in an int array, assume the base address of the array `A` is stored in the register `$s1` and the size of the array is stored in the register `$s2`. You can use some registers for storing temporary values.

One possible solution:

```
      lw $t0,0($s1)          #$t0 has the first element of A
      addi $t1,$zero,0       #$t1 stores index i starts at 0
loop: add $t1,$t1,1          # increment index i by 1
      beq $t1,$s2,done
      sll $t2,$t1,2          # $t2 has value of i*2^2
      add $t2,$t2,$s1        # form address of A[i] in $t2
      lw $t3,0($t2)          # load value of A[i] into $t3
      slt $t4,$t0,$t3        # maximum < A[i]?
      beq $t4,$zero,loop     #repeat with the original $t0
      add $t0,$t3,$zero          #$t0 stores the new maximum
      j loop
done:
```

Exercise 3: Write down the MIPS instructions for the following C++ code, assume the variable d of type char is stored in the register `$s0`. You can use some registers for storing temporary values.

```
switch (d) {
  case 'A': d = d / 2;
            break;
  case '?': d = d - d;
}
```

One possible solution:

```
#Use ASCII code for 'A' and '?'
addi $t0, $zero, 65    #$t0 stores 'A'
bne $s0, $t0, caseQuestionM    #$s0 stores d
srl $s0, $s0, 1
j switchEnd
 caseQuestionM:  addi $t0, $zero, 63
                 bne $s0, $t0, switchEnd
                 sub $s0, $s0, $s0
switchEnd:
```

Exercises

Exercise 4: Convert the following MIPS code into the corresponding C++ statements.

MIPS code:

```
        add $t0, $zero, $zero    #$t0 stores  the variable i
        addi $t2, $zero, 1       #$t2 stores the variable j
        addi $s0, $zero, 5
Loop:
        slt $t1, $t0, $s0
        beq $t1, $zero, Done
        addi $t2, $t2, 3
        slti $t3, $t2, 9
        bne $t3, $zero, ifEND
        j Done
ifEND:
        addi $t0, $t0, 1
        j Loop
Done:
```

Solution (C++ statements):

```
j = 1;
for (int i = 0; i < 5; i++) {
      j = j + 3;
     if (j > 8)
          break;
}
```