COMP2611: Computer Organization

MIPS Procedure (Solution)

Exercises

Question 1: Write down the shortest sequence of MIPS instructions for the following C++ codes, assuming each variable is stored in a different register (you name it).

```
b = a + 0x37cf0010;
```

One possible solution:

```
lui $s0, 0x37cf              #s0 stores b
addi $s0, $s0, 0x0010
add $s0, $s0, $1             #s1 stores a
```

Exercises

Question 2: Write down the shortest sequence of MIPS instructions for the following C++ codes, assuming each variable is stored in a different register (you name it).

```
b = a + 0x37cf**f346**;
```

Solution:

```
lui $s0, 0x37cf          #s0 stores b
#note: addi doesn't work here because of its sign-extension
ori $s0, $s0, 0xf346
add $s0, $s0, $1 #s1 stores a
```

Exercises

Question 1: Translate the following C++ function into a MIPS function, using the registers `$a0` and `$a1` for its parameters and the register `$v0` for its return value.

```
int equal(int p1, int p2) {
  if (p1 == p2)
      return 1;
  return 0;
}
```

Solution:

```
equal: beq $a0, $a1, true #a0 stores p1 and a1 stores p2
         addi $v0, $zero, 0
         jr $ra
true: addi $v0, $zero, 1 #note: any non-zero value is okay
         jr $ra
```

Exercises

Question 2: Write down the MIPS instructions that make the following call to the C++ function in the previous exercise, assuming the variable `b` is stored in the register `$s0`.

```
int b = equal(3, 4);
```

Solution:

```
addi $a0, $zero, 3
addi $a1, $zero, 4
jal equal
addi $s0, $v0, 0
```

Exercises

Question 3: The following C++ function takes as inputs the base address of an int array `A` and returns the minimum value in `A`. Using the registers `$a0` and `$a1` as arguments to the function, `$v0` as returned value, `$s0` as base address of `A` and `$s1` as the size of `A`, translate the C++ function into a MIPS function

```
int minArray(int A[], int arraySize) {
  int min = A[0];
  int i = 1;
  while(i < arraySize) {
     if(min < A[i])
        min = A[i];
     i++;
  }
  return min;
}
```

One possible solution:

```
minArray:
  lw $s1, 0($a0)        #$s1 stores A[0] which is initial min
  addi $s2, $zero, 1    #$s2 stores 1
loop:
  slt $t0, $s2, $a1
  beq $t0, $zero, End #$t0 = 0 if i >= arraySize
  sll $t1, $s2, 2
  add $t1, $t1, $a0
  lw $s3, 0($t1)
  slt $t2, $s1, $s3
  bne $t2, $zero, Inc #$t2 != 0 if min < A[i]
  add $s1, $s3, $zero
inc:
  addi $s2, $s2, 1
  j loop
end:
  addi $v0, $s1, 0
  jr $ra
```