King
CS 331
27 September 2020

# Homework Assignment 2

## Objectives

The purpose of this assignment is to:

- Understand MIPS instructions, formats and fields,
- Write simple assembly programs using the MIPS instruction set,
- Convert assembly instructions to C code and vice versa,
- Convert assembly instructions to binary words and vice versa.

## Guidelines

All question numbers refer to the exercises at the end of Chapter 2 of the textbook. Solutions for the following problems are to be completed independently. Type each answer and clearly show the work/steps. Make sure to write any formula you use, do not plug in numbers right away.

## Questions

**Exercise 2.3**

For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

```
B[8] = A[i–j];
```

Given:
A = s6, B = s7
i = s3, j = s4

Algorithm:
- array is in memory
- load A[i-j] into some register, manually computing address of word “A[i-j]” before lw
- store into B[8]

```
sub t0, s6, s7        # store i-j into saved temporary t0
sll t0, t0, 2         # &A[i-j] = &A[0] + 4x(i-j). Shifting left twice =
                      # multiplying by 4
add t1, s6, t0        # storing sum of A[0] and t0; t1=&A[i-j]
lw t2, 0(t1)          # load A[i-j] into temp register t2
sw, t2, 32(s7)        # store A[i-j] from t2 into B[8]. 8th word = byte 32 (8*4)
```

**Exercise 2.4**

For the MIPS assembly instructions below, what is the corresponding C statement? Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

```
sll $t0, $s0, 2           # $t0 = f * 4
add $t0, $s6, $t0         # $t0 = &A[f]
sll $t1, $s1, 2           # $t1 = g * 4
add $t1, $s7, $t1         # $t1 = &B[g]
lw $s0, 0($t0)            # f = A[f]
addi $t2, $t0, 4
lw $t0, 0($t2)
add $t0, $t0, $s0
sw $t0, 0($t1)
```

```
#t0 = f * 4
#t0 = &A[f]
#t1 = g * 4
#t1 = &B[g]
#f = A[f]

#t2 = t0 + 4 = A[f] + 4
#t0 = t2 = A[f] + 4
#t0 = t0 + f = A[f] + 4 + A[f]
#t0 = t1 = B[g]
```

**Exercise 2.16**

Provide the type, assembly language instruction, and binary representation of instruction described by the following MIPS fields:

```
op=0, rs=3, rt=2, rd=3, shamt=0, funct=34
```

- Type:
  This is an R-format instruction since there are fields defined for the opcode, rs, rt, rd, shamt, and funct fields.

- Assembly language instruction:
  op=0
  rs=3
  rt=2
  rd=3
  shamt=0
  funct=34
  ?

- Binary representation:
  000 000  000 011  000 010  000 011  000 000  100 010
  = $(000000\ 000011\ 000010\ 000011\ 000000\ 100010)_2$

**Exercise 2.17**

Provide the type, assembly language instruction, and binary representation of instruction described by the following MIPS fields:

op=0x23, rs=1, rt=2, const=0x4

- Type:
  This is an I-format instruction since there are fields defined for the opcode, rs, rt, and immediate fields.

- Assembly language instruction:
  ?

- Binary representation:
  ?

**Exercise 2.18 (Note: effect on register fields and opcode bits)**

Assume that we would like to expand the MIPS register file to 128 registers and expand the instruction set to contain four times as many instructions.

1. How would this affect the size of each of the bit fields in the R-type instructions?
2. How would this affect the size of each of the bit fields in the I-type instructions?

3. How could each of the two proposed changes decrease the size of an MIPS assembly program? On the other hand, how could the proposed change increase the size of an MIPS assembly program?

1. ?
2. ?
3. ?

**Exercise 2.19 (Notes: there is a typo in the first instruction of 2.19.1, it is sll $t2, $t0, 4. In 2.19.3, pay attention to the type of extension for andi)**

Assume the following register contents:

$t0 = 0xAAAAAAAA, $t1 = 0x12345678

1. For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
sll $t2, $t0, 4
or $t2, $t2, $t1
```

2. For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
sll $t2, $t0, 4
andi $t2, $t2, -1
```

3. For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
srl $t2, $t0, 3
andi $t2, $t2, 0xFFEF
```

1. ?
2. ?
3. ?

**Exercise 2.23**

Assume $t0 holds the value 0x00101000. What is the value of $t2 after the following instructions?

```
          slt $t2, $0, $t0
          bne $t2, $0, ELSE
          j DONE
ELSE: addi $t2, $t2, 2
DONE:
```

```
#t2 = (0 < t0)              .. 0 < 0x00101000, so t2 = 1
#goto ELSE if t2 != 0       .. 1 != 0, so goto ELSE
#goto DONE                  ..
#ELSE: t2 = t2 + 2          .. t2 = 3
#DONE:                      ..

The final value of $t2 is 3.
```

**Exercise 2.24**

Suppose the program counter (PC) is set to 0x2000 0000. Is it possible to use the jump (j) MIPS assembly instruction to set the PC to the address as 0x4000 0000? Is it possible to use the branch-on-equal (beq) MIPS assembly instruction to set the PC to this same address?

```
?
```

**Exercise 2.26**

Consider the following MIPS loop:

```
LOOP: slt $t2, $0, $t1
      beq $t2, $0, DONE
      subi $t1, $t1, 1
      addi $s2, $s2, 2
      j LOOP
DONE:
```

1. Assume that the register $t1 is initialized to the value 10. What is the value in register $s2 assuming $s2 is initially zero?
2. For each of the loops above, write the equivalent C code routine. Assume that the registers $s1, $s2, $t1, and $t2 are integers A, B, i, and temp, respectively.
3. For the loops written in MIPS assembly above, assume that the register $t1 is initialized to the value N. How many MIPS instructions are executed?

```
#LOOP: t2 = (0 < t1)
#goto DONE if t2 == 0
#t1 = t1 - 1
#s2 = s2 + 2
#goto LOOP
#DONE:
```

1. If t1=10 and s2=0, the final value in s2 is 20. Register t2 represents the sentinel value for the condition (t1 > 0), which is false after 10 iterations. So once the loop is complete, 2 has been added to register s2 10 times.
2. // assume that the variables i and B have been initialized
   // to some positive values.

   // using  a while loop:
   while (i > 0) {
        i = i - 1;
        B = B + 2;
   }
   // done

   // using a for loop:
   for ( ; i > 0; i--) {
        B = B + 2;
   }
   // done
3. If t1=N, then a total of 5*(N-1) + 2 instructions are executed. The first five lines will be executed through each iteration of the loop until the condition for the slt instruction evaluates to false (so N-1 times). The last iteration executes the first two lines only.