Islamic University – Gaza
Engineering Faculty
Department of Computer Engineering
ECOM 3010: Computer Architecture Discussion

Chapter 2

Exercises with solutions

Eng. Eman R. Habib

October, 2013

Discussion exercises

Exercise 1:

Convert the following C statements to equivalent MIPS assembly language. Assume that the variables f, g, I and j are assigened to registers $s0, $s1, $s2 and $s3 respectively. Assume that the base address of the array A and B are in registers $s6 and $s7 respectively.

a) f = g + h + B[4]

```
lw $t0, 16($s7)
add $s0, $s1, $s2
add $s0, $s0, $t0
```

b) f = g – A[B[4]]

```
lw $t0,16($s7)
sll $t1, $t0, 2
add $t2, $t1, $s6
lw $t3, 0($t2)
sub $s0, $s1, $t3
```

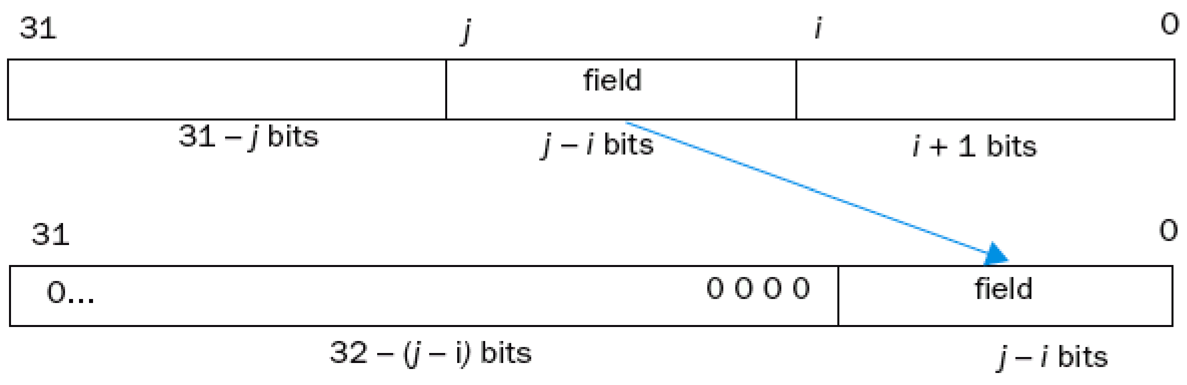
Exercise 2: (2.4 from book)

Why doesn't MIPS ha

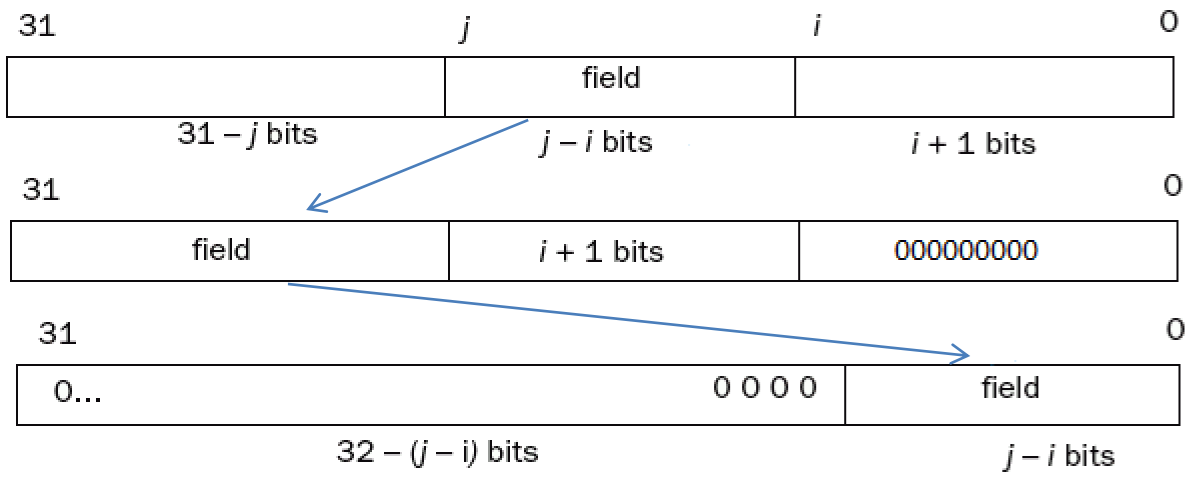
Since MIPS includes negative, its range $\pm 2^{15}$, add immediate mediate with positive number, so subtract immediate would be redundant

Exercise 3: (2.6 from book)

Find the shortest sequence of MIPS instructions that extracts a field for the constant values i=5 and j=22 from register `$t3` and places it in register `$t0`.



```
sll $t0, $t3, 9
srl $t0, $t0, 15
```



- 31-j = 31-22 = 9

```
sll $t0, $t3, 9
```

- i+1+9 = 5+1+9 = 15

```
srl $t0, $t0, 15
```

Another solution:

- i+1 = 6

```
srl $t0, $t3, 6
```

- j-i = 22-5=17

```
andi $t0, $t0,
```

Exercise 4: (2.32 from book)

Show the single MIPS instruction for this C statement

```
    b = 25 | a;
ori $t1, $t0, 25
```

Exercise 5:

Convert the MIPS instruction to machine language:

```
    srl $s1, $t2, 3
```

srl is R-type, opCode is 0 and function is 2
$s1 = 17 is rd
$t2 = 10 is rt
rs unused
shamt is 3

$\underbrace{\text{0000 00}}_{\text{Op(6)}}/\underbrace{\text{00 000}}_{\text{rs(5)}}/\underbrace{\text{0 1010}}_{\text{rt(5)}}/\underbrace{\text{1000 1}}_{\text{rd(5)}}/\underbrace{\text{000 11}}_{\text{shamt(5)}}/\underbrace{\text{00 0010}}_{\text{func(6)}}$ = 0x00A88C2

Exercise 6:

Translate the following machine code to MIPS:

$$\underbrace{1010\ 11}_{\substack{43\\ \text{Op(6)}}}/\underbrace{10\ 000}_{\substack{16\\ \text{rs(5)}}}/\underbrace{0\ 1011}_{\substack{11\\ \text{rt(5)}}}\ /\underbrace{0000\ 0000\ 0000\ 0100}_{\substack{4\\ \text{immediate(16)}}}$$

Op = 43 = sw
16 = $s0
11 = $t3

```
sw $t3, 4($s0)
```

Exercise 7: (2.37 from book)

For each pseudoinstruction in the following table, produce a minimal sequence of actual MIPS instructions to accomplish the same thing. In the following table, `big` refers to a specific number that requires 32 bits to represent and `small` to a number that can fit in 16 bits.

Pseudoinstruction	What it accomplishes	Solution
`move $t1, $t2`		`$t1, $t2, $zero`
`clear $t0`		`$t0, $zero, $zero`

`beq $t1, small`		`$t0, $zero, small`
		`1, $t0, L`

`beq $t2, big, L`	**`if ($t2 == big) g`**	`1, upper(big)`
		`1, $t1, lower(big)`
		`beq $t2, $$t1, L`

`li $t1, small`	**`$t1 = small`**	`addi $t1, $zero, small`

`li $t2, big`	**`$t2 = big`**	`lui $t2, upper(big)`
		`ori $t2, $t2, lower(big)`

`ble $t3, $t5, L`	**`if ($t3 <= $t5) go to L`**	`slt $t0, $t5, $t3`
		`beq $t0, $zero, L`

`bgt $t4, $t5, L`	**`if ($t4 > $t5) go to L`**	`slt $t0, $t5, $t4`
		`bne $t0, $zero, L`

`bge $t5, $t3, L`	**`if ($t5 >= $t3) go to L`**	`slt $t0, $t5, $t3`
		`beq $t0, $zero, L`

`addi $t0, $t2, big`	**`$t0 = $t2 + big`**	`lui $t1, upper(big)`
		`ori $t1, $t1, lower(big)`
		`add $t0, $t2, $t1`

`lw $t5, big($t2)`	**`$t5 = Memory[$t2 + big]`**	`lui $t1, upper(big)`
		`ori $t1, $t1, lower(big)`
		`add $t1, $t1, $t2`
		`lw $t5, 0($t1)`

Exercise 8:

Convert the following C fragment to equivalent MIPS assembly language. Assume that the variables a and b are assigened to registers $s0 and $s1 respectively. Assume that the base address of the array D is in register $s2.

```
while(a < 10){
     D[a] = b + a;
     a += 1;
}
Loop: stli $t0, $s0, 10
      beq $t0, $zero, exit
      sll $t1, $s0, 2
      add $t1, $t1, $s2
      add $t2, $s1, $s0
      sw $t2, 0($t1)
      addi $s0, $s0, 1
      j Loop
exit:
```

Exercise 9:

Show the effects on memory and registers of the following instructions. Suppose a portion of memory contains the

0x10000000	0
0x10000004	0

And register **$t0 contains 0x10000000 and $s0 contains 0x01234567.** Assume each of the following instructions is executed independently of the others, starting with the values given above. Hint: Don't forget that the MIPS architecture is Big-Endian.

The memory:

Address	Data
0x10000000	0x12
0x10000001	0x34
0x10000002	0x56
0x10000003	0x78
0x10000004	0x9A
0x10000005	0xBC
0x10000006	0xDE
0x10000007	0xF0

a) lw $t1, 0($t0)
   $t1 = 0x12345678
b) lw $t2, 4($t0)
   $t2 = 0x9ABCDEF0
c) lb $t3, 0($t0)
   $t3 = 0x00000012
d) lb $t4, 4($t0)
   $t4 = 0xFFFFFF9A → lb is sign extended
e) lb $t5, 3($t0)
   $t5 = 0x00000078
f) lh $t6, 4($t0)
   $t6 = 0XFFFF9ABC → lh is sign extended
g) sw $s0, 0($t0)
   at address 0x10000000 will contain 0x01234567

Address	Data
0x10000000	0x01
0x10000004	
0x10000005	
0x10000006	
0x10000007	0xF0

h) sb $s0, 4($t0)
   the address 0x10000004 will contain 0x67BCDEF0

Address	Data
0x10000000	0x12
0x10000001	0x34
0x10000002	0x56
0x10000003	0x78
0x10000004	0x67
0x10000005	0xBC
0x10000006	0xDE
0x10000007	0xF0

i) sb $s0, 7($t0)

the address 0x10000004 will contain 0x9ABCDE67

Address	Data
0x10000000	0x12
0x10000001	0x34
0x10000002	0x56
0x10000003	0x78
0x10000004	0x9A
0x10000005	0xBC
0x10000006	0xDE
0x10000007	0x67

Exercise 10:

Convert the following program into machine code.

```
0xFC00000C      start: ……………
0xFC000010      loop: addi $t0,$t0,-1
0xFC00001
0xFC00001
0xFC00001
```

```
addi $t0,$t0,-1
-1 = 0xFFFF
```

$$\underbrace{\text{0010 00}}_{\substack{8\\ \text{Op(6)}}}/\underbrace{\text{01 000}}_{\substack{8\\ \text{rs(5)}}}/\underbrace{\text{0 1000}}_{\substack{8\\ \text{rt(5)}}}/\underbrace{\text{1111 1111 1111 1111}}_{\substack{-1\\ \text{immediate(16)}}}$$

```
sw $t0, 4($t2)
```

$$\underbrace{\text{1010 11}}_{\substack{43\\ \text{Op(6)}}}/\underbrace{\text{01 010}}_{\substack{10\\ \text{rs(5)}}}/\underbrace{\text{0 1000}}_{\substack{8\\ \text{rt(5)}}}/\underbrace{\text{0000 0000 0000 0100}}_{\substack{4\\ \text{immediate(16)}}}$$

```
bne $t0, $t3, loop
```

target address = (immediate * 4) + address of the following instruction

immediate = (target address – address of the following instruction) / 4

= (FC000010 – FC00001C) / 4

= - C /4

= -3 → 1111 1111 1111 1101

Or convert to bainary first
= 1111 1100 0000 0000 0000 0000 0001 0000 –
1111 1100 0000 0000 0000 0000 0001 1100

= 1111 1100 0000 0000 0000 0000 0001 0000 +
0000 0011 1111 1111 1111 1111 1110 0100

= 1111 1111 1111 1111 1111 1111 1111 0100 / 4 → srl by 2
= 11 1111 1111 1111 1111 1111 1111 1101

immediate is 16 only so immediate = 1111 1111 1111 1101

$$\underbrace{\text{0001 01}}_{\substack{5\\ \text{Op(6)}}}/\underbrace{\text{01 000}}_{\substack{8\\ \text{rs(5)}}}/\underbrace{\text{0 1011}}_{\substack{11\\ \text{rt(5)}}}\ /\underbrace{\text{1111 1111 1111 1101}}_{\substack{-3\\ \text{immediate(16)}}}$$

```
j start
```

target address = last

immediate = first 28

= C00000C / 4 = 3000003

= 1100 0000 0000 0000 0000 0000 1100 / 4 → srl b

= 11 0000 0000 0000 0000 0000 0011

$$\underbrace{\text{0000 10}}_{\substack{2\\ \text{Op(6)}}}/\underbrace{\text{11 0000 0000 0000 0000 0000 0011}}_{\substack{3000003\\ \text{immediate (26)}}}$$

Exercise 11: (2.38 from book)

Explain why an assembler might have problems directly implementing the branch instruction in the following code sequence:

```
here: beq $s0, $s2, there
…
there:  add $s0, $s0, $s0
```

Show how the assembler might rewrite this code sequence to solve these problems.

The problem is that we are using PC-relative addressing, so if that address is too far away, we won't be able to use 16 bits to describe where it is relative to the PC.

If `there` refers to a location further than 128 KB from the PC, the solution would be:

```
here:     bne $s0, $s2, skip
          j there
skip:
…
there:    add $s0, $s0, $s0
```

If `there` refers to a location further than 256 MB from the PC, the solution would be:

```
here:     bne $s0, $s2, skip
          lui $ra, there(upper)
          ori $ra, $ra there(lower)
          jr $ra
skip:
…
there:    add $s0, $s0, $s0
```

Exercise 12:

Suppose that you ha ng signature:

```
int sum(int A[], int first, int las
```

This function calculates the sum of the elements of A nt first and ending with element last. Write a fragment of MIPS assembly language which calls this function and uses it to calculate the average of all values in A. You may assume that the size of the array A is N, the base address of A in $a0.

```
Average:
     add $a1, $zero, $zero      # index of first element
     addi $a2, $zero, N
     addi $a2, $a2, -1          # index of last element is N-1
     jal sum
     add $t0, $zero, $v0        # Save the return value in $t0
     addi $t1, $zero, N         # Load size of array into $t1
     div $t2, $t0, $t1          # This form of div is provided as
                                a  pseudoinstruction.
```

Exercise 13:

Below is a recursive version of the function BitCount. This function counts the number of bits that are set to 1 in an integer.
Your task is to translate this function into MIPS assembly code. The parameter x is passed to your function in register $a0. Your function should place the return value in register $v0.

```
int BitCount(unsigned x) {
int bit;
if (x == 0)
 return 0;
bit = x & 0x1;
return bit + BitCount(x >> 1);
}
```

```
BitCount:
     addi $sp, $sp, -8
     sw $s0, 4($sp)
     sw $ra, 0($sp)
     bne $a0, $0, else
     add $v0, $0, $0
     addi $sp, $sp, 8
     jr $ra
else:
     andi $s0, [illegible]
     srl $a0, $a0, 1
     jal BitCount
     add $v0, $v0, $s0
     lw $ra, 0($sp)
     lw $s0, 4($sp)
     addi $sp, $sp, 8
     jr $ra
```

Extra exercises

Exercise 14: (2.29 from book)

Add comments to the following MIPS code describe in one sentence what it computes. Assume that $a0 and $a1 are used for the input and both initially contain the integers a and b, respectively. Assume that $v0 used for the output.

```
          add $t0, $zero, $zero  # initialize running sum $t0 = 0
loop:     beq $a1, $zero, finish # finished when $a1 is 0
          add $t0, $t0, $a0      #compute running sum of $a0
          sub $a1, $a1, 1        # compute this $a1 times
          j loop
finish:   addi $t0, $t0, 100     # add 100 to a * b
          add $v0, $t0, $zero    # return a * b + 100
```

```
t0=0
while(a1 != 0){
     t0 = t0 + a0;
     a1 = a1 – 1;
}
t0 = t0 + 100;
v0 = t0;
```

The program computes a * b + 100.

Exercise 15: (2.34 from book)

The following program tries to copy words from the address in register $a0 to the address in register $a1, counting the number of words copied in register $v0. The program stops copying when it finds a word equal to 0. You do not have to preserve the contents of registers %v1, $a0 and $a1. This terminating word should be copied but not counted.

```
          addi $v0, $zero, 0 # Initialize count
     loop:lw, $v1, 0($a0) # Read next word from source
          sw $v1, 0($a1) # Write to destination
          addi $a0, $a0, 4 # Advance pointer to next source
          addi $a1, $a1, 4 # Advance pointer to next destination
          beq $v1, $zero, loop # Loop if word copied != zero
```

There are multiple bugs in this MIPS program; fix them and turn in a bug-free version.

Bug 1: Count ($v0) is initialized to zero, not –1 to avoid counting zero word.
Bug 2: Count ($v0) is not incremented.
Bug 3: Loops if word copied is equal to zero rather than not equal.

Bug-free version:

```
    addi $v0, $zero, -1 # Initialize to avoid counting zero word
loop:lw, $v1, 0($a0) # Read next word from source
     addi $v0, $v0, 1 # Increment count words copied
     sw $v1, 0($a1) # Write to destination
     addi $a0, $a0, 4 # Advance pointer to next source
     addi $a1, $a1, 4 # Advance pointer to next destination
     bne $v1, $zero, loop # Loop if word copied != zero
```

Exercise 16:

Convert the following C fragment to equivalent MIPS assembly language. Assume that the variables a, b, c, d, i and x are assigened to registers $t1, $t2, $t3, $t4, $s0 and $s1 respectively. Assume that the base address of the array A and B is in register $a0 and $a1 respectively.

```
     a) if ((a<b) && (c==0)) d = 1;

        slt $t0, $t1, $t2
        beq $t0, $0, not        #if (a>=b) go to not
        bne $t3, $0, not        #if (c!=0) go to not
        addi $t4, $0, 1
not:
```

```
     b) if (a >
        b = a +
        else
        b = a - 10;
```

```
        slt $t0, $0, $t1 # if $0 < $                    else $t0 = 0
        beq $t0, $0, else  # if $t0 == $0 then branch to else
        addi $t2, $t1, 10
        j exit
else:  addi $t2, $t1, -10
exit:

     c) A[x+3] = B[x+2] | 0x10

        addi $t0, $s1, 2 # $t0 = x+2
        sll $t0, $t0, 2 # $t0 = (x+2)*4
        add $t1,$a1,$t0 # $t1 = (base address of B + (x + 2))
        lw $t2,0($t1) # $t2 = B[x+2]
        ori $t3,$t2,0x10 # $t3 = B[x+2] | 0x10
        addi $t4,$s0,3 # $t4 = x+3
        sll $t4, $t4, 2 # $t4 = (x+3)*4
        add $t5,$a0,$t4 # $t5 = (base address of A + (x + 3))
        sw $t3,0($t5) # A[x+3] = $t3
        d) for(int i=0; i<5; i++){
              a += b;
           }

             add $s0, $zero, $zero
       Loop: stli $t0, $s0, 5
             beq $t0, $zero, exit
             add $t1, $t1, $t2
             addi $s0, $s0, 1
             j Loop
       exit:
```

Exercise 17:

Convert this high level language code into MIPS code. Do not forget to write MIPS code for the abs(x) procedure. (i saved in $s0, the address of a in $s1, y in $s2)

```
     i = 3;
     y = y + abs(a[i]);

          addi $s0,$zero,3
          sll $t0,$s0,2
          add $t1,$t0,$s1
          lw $
          jal
          add
          j exitall
     abs:
          slt $t0,$a0,$zero
          bne $t0,$zero,else
          add $v0,$a0,$zero
          jr $ra
     else:
          sub $v0,$zero,$a0
          jr $ra
     exitall:
```

Exercise 18:

For each of the following, write the shortest sequence of MIPS assembly instructions to perform the specified operation.

(Hint: $12345678 = 188 \times 2^{16} + 24910$
$31415924 = 479 \times 2^{16} + 24180$)

a) $v0 = 12345678

```
     lui $v0, 188
     ori $v0, $vo, 24910
```

b) if ($t0 < 12345678) go to address less

```
lui  $t1, 188
ori  $t1, $t1, 24910
slt  $t2, $t0, $t1
bne  $t2, $zero, less
```

c) t1 = 12345678 + 31415924

```
lui  $t0, 188
ori  $t0, $t0, 24910
lui  $t2, 479
ori  $t2, $t2, 24180
add  $t1, $t0, $t2
```

Exercise 19:

Given the register values in two's complement representation

$s1 = 0000 0000 0000 0000 0000 0000 0000 0101 = 5

$s2 = 0000 0000 0000 0000 0000 0000 0000 0011 = 3

$s3 = 1111 1111 1111 1111 1111 1111 1111 1100 = -4

What are the values of registers $s1 through $s4 after executing the following MIPS instructions:

```
slt  $s1,
slt  $s2,
sltu $s3,
sltu $s4, $s1, $s3        Resul
```

sltu compares the unsigned values in [illegible], so in the last instruction $s1 less than $s3 [illegible] big unsigned value, so $s4 = 1.

☺ *Best Wishes* ☺