COE 301 / ICS 233
Computer Organization

Major Exam 1 – Spring 2017

Saturday, March 18, 2017

10 AM – 12 Noon

Computer Engineering Department
College of Computer Sciences & Engineering
King Fahd University of Petroleum & Minerals

Student Name: _____

Student ID: _____

Section: _____

Q1	/ 20	Q2	/ 12
Q3	/ 13	Q4	/ 20
Q5	/ 20	Q6	/ 20
Total	/ 105		

Important Reminder on Academic Honesty

Using unauthorized information or notes on an exam, peeking at others work, or altering graded exams to claim more credit are severe violations of academic honesty. Detected cases will receive a failing grade in the course.

Prepared by Dr. Muhamed Mudawar and Dr. Marwan Abu-Amara

Question 1: Fill-in the Blanks

a) (2 pts) Imagine that you are working for a company that fabricates a certain IC chip. The cost per wafer is $3000, and each wafer has 2000 dies. If the cost of a good die is $2.50, then the yield of this manufacturing process is _____.

b) (2 pts) Given that the instruction **`j NEXT`** is at address **`0x004000F4`**, and the label **`NEXT`** is at address **`0x00402AEC`**. Then, the 26-bit immediate stored in the jump instruction for the label **`NEXT`** is _____.

c) (3 pts) Given the following data definitions, the address of the first variable **X** is given at **`0x10010000`** (hexadecimal), the hexadecimal addresses for **Y**, **Z**, and **S** will be:

```
.data
X: .half   1, 2, 3
Y: .byte   'A', 'B', 'C'
Z: .word   7, 8, 9
.ALIGN 3
S: .asciiz  "STRING"
```

Address of Y = _____,

Address of Z = _____,

Address of S = _____.

d) (3 pts) Show the MIPS assembly language instruction that is equivalent to the following machine language instruction. Provide the immediate value in **decimal**. The MIPS Reference data sheet is attached at the end.

Machine language instruction	MIPS assembly language instruction
`0011 0001 0001 0001 1000 0111 0110 0101`	

e) (5 pts) Each square in the table shown below represents one byte in memory and each row stores 8 bytes in memory. Starting at address **0×10010000** in the data segment, show the **byte content** in memory in **hexadecimal** for the following data definitions. If a byte is not used (or uninitialized) then leave it empty. Fill only the bytes that are initialized. For words and half words, the little endian byte ordering should be used.

```
.DATA
.WORD       -2
.HALF       0×1FFF
.ALIGN      2
.BYTE       11:3
.ALIGN      4
.BYTE       13, -1
```

Address	+0	+1	+2	+3	+4	+5	+6	+7
0×10010000								
0×10010008								
0×10010010								
0×10010018								

f) (5 pts) Given the following contents of memory, where each square represents only one byte in memory, show the values of registers **$t0** thru **$t4** in **hexadecimal** after executing each of the following MIPS assembly language instructions. The little endian byte ordering should be used. Assume **$s0 = 0×10010020**.

Address	+0	+1	+2	+3	+4	+5	+6	+7
0×10010020	0xFA	0x20	0x10	0xC0	0xB0	0x5F	0x94	

lw $t0, 0($s0)	$t0 =
lh $t1, 2($s0)	$t1 =
lhu $t2, 4($s0)	$t2 =
lb $t3, 5($s0)	$t3 =
lbu $t4, 6($s0)	$t4 =

Question 2: Pseudo-Instructions

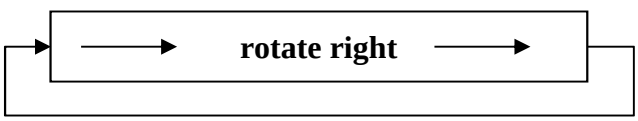
For each of the following pseudo-instructions, produce a **minimal** sequence of basic MIPS instructions to accomplish the same thing. You may use the **`$at`** register only as a temporary register.

a) `abs $t1, $t2  # absolute value` **(3 pts)**

b) `addiu $t1, $t2, 0x1234abcd  # 32-bit constant` **(3 pts)**

c) `bgt  $t1, 100, Label  # branch if greater than 100` **(3 pts)**

d) `ror  $t1, $t2, 15  # rotate right value of $t2 15 bits` **(3 pts)**



Question 3: Trace the Execution of the following Code

a) (6 pts) Given that **`Array`** is defined as shown below, determine the content of registers **`$v0`** and **`$v1`** after executing the following code. **Explain what the program is doing**.

```
Array: .word 15, -19, 17, 20, -10, 12, 100, -5

        la   $a0, Array # $a0 = 0x10010000
        addi $a1, $a0, 28
        move $v0, $a0
        lw   $v1, 0($v0)
        move $t0, $a0
loop:   addi $t0, $t0, 4
        lw   $t1, 0($t0)
        bge  $t1, $v1, skip
        move $v0, $t0
        move $v1, $t1
skip:   bne  $t0, $a1, loop
```

b) (7 pts) Given that **`Array`** is defined as shown below, determine the content of **`Array`** after executing the following code. **Explain what the program is doing.**

```
Array: .half 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

      la   $a0, Array
      li   $a1, 6
      move $t0, $a0
      addi $t1, $a0, 12

loop: lh   $t3, ($t0)
      lh   $t4, ($t1)
      sh   $t3, ($t1)
      sh   $t4, ($t0)
      addi $t0, $t0, 2
      addi $t1, $t1, 2
      addi $a1, $a1, -1
      bne  $a1, $zero, loop
```

Question 4: Writing MIPS code

a) (10 pts) Write a MIPS loop to count the number of **1's** in register **`$a0`** and put the result in register **`$v0`**. For example, if **`$a0 = 0xffff0000`** then **`$v0 = 16`**.

b) (10 pts) Write a MIPS loop that converts a string to lower case. The address of the string exists in register **`$a0`**. The string is terminated with a null character. The string should be converted and stored in memory. Check each character if it is an upper case letter (range 'A' to 'Z') before converting it to lower case. Recall that **'A' = 0x41** and **'a' = 0x61**.

Question 5: Translating Nested Loops into MIPS Assembly Language

(20 pts) Translate the following nested loops into MIPS assembly language. Given that **$a0** = number **n** of elements in all arrays, **$a1** = address of the array **a[]**, **$a2** = address of the array **b[]**, and **$a3** = address of the array **c[]**. Each array element is a 32-bit signed integer. Insert comments to clarify the meaning of instructions and the use of registers.

```
for (i=0; i != n; i++) {
   int cnt = 0;
   for (j=0; j != n; j++) {
      if (a[i] == b[j]) cnt = cnt + 1;
   }
   c[i] = cnt;
}
```

Question 6: The Transposition of a Matrix

(20 pts) Transposition is an important matrix operation. Given that matrix **A** is a square matrix of integers with dimensions **n×n**, the transposition is accomplished by swapping matrix element **A[i][j]** with element **A[j][i]**, as shown in the following nested loops. Given that register **$a0=n**, and register **$a1** = **address** of matrix **A**, translate the following nested loops into MIPS assembly language code.

```
for (i=0; i != n; i++) {
   for (j=i+1; j != n; j++) {
      temp1 = A[i][j];
      temp2 = A[j][i];
      A[i][j] = temp2;
      A[j][i] = temp1;
   }
}
```

Additional Page if Needed