COE 308 – Computer Architecture

Exam I – Spring 2008

Tuesday, April 1st, 2008

7:00 pm – 9:00 pm

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

Student Name: _____

Student ID: _____

Q1	/ 15	Q2	/ 15
Q3	/ 15	Q4	/ 10
Q5	/ 10	Q6	/ 20
Q7	/ 20		
Total	/ 105		

Important Reminder on Academic Honesty

Using unauthorized information 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.

Q1. (15 pts) Given the bit pattern:

`1100 0110 1101 0100 0000 0000 0000 0000` (binary)

What is the decimal value of the above number, assuming it is

a) (2 pts) Unsigned integer?

b) (2 pts) Signed integer?

c) (5 pts) Single-precision floating-point number?

d) (6 pts) Show the **Single precision** IEEE 754 representation for **`-0.05`**, rounded to the nearest even.

Q2. (15 pts) Consider the following data definitions:

```
.data
var1:    .byte     3, -2, 'A'
var2:    .half     1, 256, 0xffff
var3:    .word     0x3de1c74, 0xff
.align 3
str1:    .asciiz   "COE308"
```

a) Show the content of each byte of the allocated memory, **in hexadecimal** for the above data definitions. The **Little Endian** byte ordering is used to order the bytes within words and halfwords. Fill the symbol table showing **all labels** and their **starting address**. The ASCII code of character 'A' is 0x41, and '0' is 0x30. Indicate which bytes are skipped or unused in the data segment.

Data Segment

Address	Byte 0	Byte 1	Byte 2	Byte 3
0x10010000	0x03			
0x10010004				
0x10010008				
0x1001000C				
0x10010010				
0x10010014				
0x10010018				
0x1001001C				
0x10010020				
0x10010024				
0x10010028				
0x1001002C				

Symbol Table

Label	Address
var1	**0x10010000**

b) How many bytes are allocated in the data segment including the skipped bytes?

Q3. (15 pts) For each of the following pseudo-instructions, produce a **minimal** sequence of real MIPS instructions to accomplish the same thing. You may use the **`$at`** register only as a temporary register.

a) `abs $s1, $s2`

b) `addiu $s1, $s2, imm32  # imm32 is a 32-bit immediate`

c) `bleu $s1, $s2, Label  # branch less than or equal unsigned`

d) `bge $s1, imm32, Label  # imm32 is a 32-bit immediate`

e) `rol $s1, $s2, 5  # rol = rotate left $s2 by 5 bits`

Q4. (10 pts) Translate the following loop into assembly language where **a** and **b** are integer arrays whose base addresses are in **$a0** and **$a1** respectively. The value of **n** is in **$a2**.

```
for (i=0; i<n; i++) {
  if (i > 2) {
    a[i] = a[i-2] + a[i-1] + b[i];
  }
  else {
    a[i] = b[i]
  }
}
```

Q5. (10 pts) Translate the following **if-else** statement into assembly language:

```
if (($t0 >= '0') && ($t0 <= '9')) {$t1 = $t0 – '0';}
else if (($t0 >= 'A') && ($t0 <= 'F')) {$t1 = $t0+10-'A';}
else if (($t0 >= 'a') && ($t0 <= 'f')) {$t1 = $t0+10-'a';}
```

Q6. (20 pts) Given that x = **1 10000101 10110000000000000000001$_2$** and y = **1 01111111 01000000000000011000000$_2$** are single precision IEEE 754 floating-point numbers. Perform the following operations showing all the intermediate steps and final result in binary. Round to the nearest even.

a) (10 pts) $x + y$

b) (10 pts) $x * y$

Q7. (20 Pts) Write MIPS assembly code for the procedure **BinarySearch** to search an array which has been previously sorted. Each element in the array is a 32-bit signed integer. The procedure receives three parameters: register **$a0** = **address of array** to be searched, **$a1** = **size** (number of elements) in the array, and **$a2** = **item** to be searched. If found then **BinarySearch** returns in register **$v0** = **address** of the array element where **item** is found. Otherwise, **$v0** = **0**.

```
BinarySearch ($a0=array, $a1=size, $a2=item) {
  lower = 0;
  upper = size-1;
  while (lower <= upper) {
    middle = (lower + upper)/2;
    if (item == array[middle])
      return $v0 = ADDRESS OF array[middle];
    else if (item < array[middle])
      upper = middle–1;
    else
      lower = middle+1;
  }
  return $v0=0;
}
```

Additional Page if Needed