ICS 233 - Computer Architecture & Assembly Language

Exam I – Fall 2007

Saturday, November 3, 2007
7:00 – 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	/ 15
Q7	/ 20		
Total	/ 100		

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.

Q1. (15 pts) Find the word or phrase that best matches the following descriptions:

a) Program that manages the resources of a computer for the benefit of the programs that run on that machine.

b) Program that translates from a high-level notation to assembly language.

c) Component of the processor that tells what to do according to the instructions.

e) Interface that the hardware provides to the software.

d) Microscopic flaw in a wafer.

f) Rectangular component that results from dicing a wafer.

g) Computer inside another device used for running one predetermined application or collection of software.

h) (3 pts) In a magnetic disk, the disks containing the data are constantly rotating. On average, it should take half a rotation for the desired data on the disk to spin under the read/write head. Assuming that the disk is rotating at 10000 RPM (Rotations Per Minute), what is the average time for the data to rotate under the disk head?

i) (5 pts) Assume you are in a company that will market a certain IC chip. The cost per wafer is $5000, and each wafer can be diced into 1200 dies. The die yield is 40%. Finally, the dies are packaged and tested, with a cost of $9 per chip. The test yield is 80%; only those that pass the test will be sold to customers. If the retail price is 50% more than the cost, what is the selling price per chip?

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   "ICS233"
```

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. The following code fragment processes two arrays and produces an important result in register **$v0**. Assume that each array consists of **N** words, the base addresses of the arrays **A** and **B** are stored in **$a0** and **$a1** respectively, and their sizes are stored in **$a2** and **$a3**, respectively.

```
        sll    $a2, $a2, 2
        sll    $a3, $a3, 2
        addu   $v0, $zero, $zero
        addu   $t0, $zero, $zero
outer:  addu   $t4, $a0, $t0
        lw     $t4, 0($t4)
        addu   $t1, $zero, $zero
inner:  addu   $t3, $a1, $t1
        lw     $t3, 0($t3)
        bne    $t3, $t4, skip
        addiu  $v0, $v0, 1
skip:   addiu  $t1, $t1, 4
        bne    $t1, $a3, inner
        addiu  $t0, $t0, 4
        bne    $t0, $a2, outer
```

a) (5 pts) Describe what the above code does and what will be returned in register **$v0**.

b) (10 pts) Write a loop that calculates the first *N* numbers in the Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, …), where *N* is stored in register **$a0**. Each element in the sequence is the sum of the previous two. Declare an array of words and store the generated elements of the Fibonacci sequence in the array.

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