CS/ECE 333 Test #1
Fall Semester – 2006 -- SOLUTION
September 29, 2006

Please write your answers on the test pages. Please write clearly, and make sure that the answer is clearly in the space provided.

Pledge:

Name: ____________________________________ Date: ________________

Please print your name: ____________________________________________

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>MAX</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td>10 EC</td>
</tr>
</tbody>
</table>

Page 1 of 8
1. (5 points) Name the special purpose register that stores the address of the next instruction to be fetched from memory.

All or none
Answer: _______________ PC or program counter _______________

2. (5 points) Name the special purpose register that is used to store the instruction while it is decoded and executed by the processor.

All or none
Answer: ___________ IR or instruction register ________________

3. (5 points) How many 32-bit words of memory can the 24-bit address identify uniquely, if memory is byte-addressable? (Assume that a byte is 8 bits.)

All or none
Answer: ______________ $2^{24}/2^2 = 2^{22}$ __________________

4. (5 points) List four of the five major components that are found in all computers.

Answers: Input, Output, Memory, Datapath (ALU), Control

1 point each

(5 points) Suppose that a computer has 128 Megabytes of memory. Exactly, how many 32-bit words of memory is this?

Answer: __________ $2^{27}/2^2 = 2^{25}$ ______________________

-2 if converted incorrectly, but general idea is correct.
-2 if not base 2

5. (5 points) Explain the concept of a stored program computer.

A stored program computer is one in which the machine language program is stored in memory along with data, and the computer is able to manipulate both program and data in identical ways.

-2.5 for not mentioning program
-2.5 for not mentioning data
6. (10 points) Draw a block diagram that illustrates the concept of a memory hierarchy. What is the fastest and most expensive memory in your hierarchy? What is the slowest and least expensive memory in your hierarchy?

Fastest, Most Expensive Memory _________ Registers

Slowest, Least Expensive Memory _________ Tape

MEMORY HIERARCHY BLOCK DIAGRAM

<table>
<thead>
<tr>
<th>Fastest</th>
<th>Slowest</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Registers, Cache, Main memory, Disk memory, Tape memory</td>
<td></td>
</tr>
</tbody>
</table>

-2 for each missing or out of place piece of memory hierarchy
-1 pt for fastest or slowest incorrect

7. (10 points) Write assembly language code for a 2-address machine and a 3-address machine to implement the expression F = (A*(B+C) + (D*E)). The 3-address machine uses instructions such as ADD X, Y, Z which adds Y and Z and places the result in X. You also have a multiply instruction given by MULT X, Y, Z which multiplies Y and Z and places the result in X. The 2-address machine has an addition instruction given by ADD X, Y which adds X and Y and places the result in X. Similarly, MULT X, Y will multiply X and Y and place the result in X. Your program must not modify the values of the variables on the right hand side of the equation. You may define temporary variables as needed.

<table>
<thead>
<tr>
<th>Two-Address Program</th>
<th>Three-Address Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD TEMP1, C</td>
<td>ADD TEMPI, B, C</td>
</tr>
<tr>
<td>ADD TEMP1, B</td>
<td>MULT TEMPI, A, TEMPI1</td>
</tr>
<tr>
<td>MULT TEMP1, A</td>
<td>MULT TEMP2, D, E</td>
</tr>
<tr>
<td>ADD TEMP2, D</td>
<td>ADD F, TEMP1, TEMP2</td>
</tr>
<tr>
<td>MULT TEMP2, E</td>
<td></td>
</tr>
<tr>
<td>ADD F, TEMP2</td>
<td></td>
</tr>
</tbody>
</table>

-1 pt each line of code incorrect
8. (15 points) Consider a computer with 8 32-bit general-purpose registers R1 through R8. All instructions in this computer are of the form:

OPERATION OPERAND1, OPERAND2

This should really read that destination is always operand 1. Announced in class.

OPERATION is a function such as ADD, MOVE, and MULT. OPERAND1 is one operand, and OPERAND2 is a second operand. The destination operand is always OPERAND2. For example, the instruction ADD R1, R2 will add the contents of the registers R1 and R2 and will place the result in register R2.

Consider the following initial contents of registers (all numbers are in decimal):

- Register R1 contains 10
- Register R2 contains 4
- Register R3 contains 8
- Register R4 contains 5

Suppose the computer executes the following simple program:

ADD R1, R4
MULT R4, R2
MOVE R2, R3
ADD R4, R3

Show in the table below the content of each register after the execution of each individual instruction.

<table>
<thead>
<tr>
<th>Register or Memory Contents</th>
<th>Initial Contents</th>
<th>ADD R1, R4</th>
<th>MULT R4, R2</th>
<th>MOVE R2, R3</th>
<th>ADD R4, R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>R2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>R3</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>R4</td>
<td>5</td>
<td>5</td>
<td>20</td>
<td>20</td>
<td>28</td>
</tr>
</tbody>
</table>

-1 point for each incorrect register contents
9. (15 points) Addressing modes provide different ways of calculating the memory addresses of operands in memory. This calculated address is called the effective address. Describe the following addressing modes and give an example using SRC instructions.
   a. Direct addressing
      i. Description: instruction contains address of operand
         ii. Example: __________________ld ra, c2_______________________
   b. Displacement addressing:
      i. Description: address of operand is register + constant
         ii. Example: __________________ld ra, c2(rb)_______________________
   c. Relative addressing:
      i. Description: address of operand is PC + constant
         ii. Example: __________________ldr ra, c1_______________________

10. (5 points) The Motorola 68000 has many addressing modes. Name the addressing mode which corresponds to the following M68k instructions:

    a. MOVE.B #12, d1
       Description: Moves byte-sized constant 12 to data register 1
       i. Addressing Mode: _________Immediate addressing____________
    b. MOVE (A6), d5
       Description: Moves operand at address in A6 to data register 5
       i. Addressing Mode: _________Register indirect_________________
11. (15 points) Most machines make use of a stack data structure to keep track of procedure activation records. Activation records contain arguments passed to a called function, local variables, and return values generated by the called function. To enable the communication between the calling function and the called function to operate correctly, there must be a calling convention or standard. Describe the responsibilities of the calling function (Caller) and the called function (Callee) as discussed in class. You may assume that there is a special purpose register called SP that contains the value of the stack pointer.

a. Caller:
   i. Stores arguments onto stack (2 pts)
   ii. Allocates space for output values (2 pts)
   iii. Calls procedure (1 pt)

b. Callee:
   i. Stores SP (1 pt)
   ii. Stores return address (2 pts)
   iii. Retrieves passed arguments (2 pts)
   iv. Saves results (no need to say performs procedure body) (2 pts)
   v. Restores SP and returns to caller (2 pts)
Extra Credit (10 pts.):

In class we showed how to write assembly code for if statements and do-while loops and discussed the steps for writing assembly for a while loop which involved converting the while loop in high-level source into an if statement whose body contains a do-while loop and then converting to assembly code.

Write SRC assembly code for the following high-level source code:

```plaintext
count = 6;
while (count > 0)
{
    count--;
}
```

Below is the sample SRC code from class for an if statement and for a do-while loop.

High-level source code:

```plaintext
if(x > 0)
    x = x + 1;
```

SRC:

```plaintext
zero: .dc 0 ; define constant zero for convenience
x: .dc 1 ; define x = 1
ld r1, x  ; load x into r1
ld r0, zero ; load zero into r0
la r4, End ; load addr of End into
```

```
addi r3, r0, r1 ; r3 ← r0 - r1
brpl r4, r3 ; br to addr in r4 if r3>0
addi r1, r1, 1 ; increment r1
st r1, x ; store r1 to x
```

End: stop

High-level source code:

```plaintext
count = 6
do{
    count--;
} while (count > 0);
```

SRC:

```plaintext
zero: .dc 0
count: .dc 6
; load values and addresses to respective registers r0, r1, r4, r5
ld r0, zero
ld r1, count
la r4, End
la r5, Loopstart
```

```plaintext
r4
sub r3, r0, r1 ; r3 ← r0 - r1
brpl r4, r3 ; br to addr in r4 if r3>0
addi r1, r1, 1 ; increment r1
st r1, x ; store r1 to x
```

Loopstart:

```plaintext
addi r1, r1, -1
st r1, count
sub r3, r0, r1
brmi r5, r3
```

```
br r4
```

End: stop
Answer:

zero: .dc 0
count: .dc 6

; load some constants and some addr (2 pts)
ld r0, zero
ld r1, count
la r2, Loopstart
la r3, End

; if statement (3 pts:
comparison + branch direction/sense + appropriate target)
sub r4, r0, r1  ; negative if count > 0
brpl r3, r4  ; go to end if count <=0

; do-while
Loopstart:
addi r1, r1, -1  ; decrement count (2 pts)
st r1, count  ; store count (1 pt)
sub r4, r0, r1  ; negative if count > 0
brmi r2, r4  ; go to loopstart if count > 0 (2pts)
br r3  ; go to End, actually not necessary

End:  stop