The 6 week exam will cover everything assigned or discussed in class from:

1. Chapter 1
2. Chapter 2
3. Appendix B
4. Any extra material in the lecture notes

Below are sample questions – not a complete set of topics.

PRACTICE IC220 Computer Architecture PRACTICE
6-Week Exam
14 Feb 2007

Name _____ Alpha _____

Section: 2001 3001 4001

Note: This exam is closed-book, closed-notes.
No calculators are permitted.
To receive partial credit, show all work.
UNLESS STATED OTHERWISE, PSEUDO-INSTRUCTIONS MAY BE USED

WORK QUICKLY AND DO THE EASY PROBLEMS FIRST!

Page 1 (pts) _____

Page 2 (pts) _____

Page 3 (pts) _____

Page 4 (pts) _____

Page 5 (pts) _____

Page 6 (pts) _____

TOTAL() _____

(1 pt) What is the program that translates from a high level language to assembly language?

(5 pts) List the five classic components associated with all computers. No description needed.

1.

2.

3.

4.

5.

Computer manufacturers have often built multiple different machines that were all based on the same "Instruction Set Architecture" (ISA).
(2 pts) What is an advantage of using the same ISA for different machines?

(2 pts) What is a disadvantage of using the same ISA for different machines?

(2 pts) Translate the following C++ expression into MIPS instruction(s).
Assume that A is $s0, B is $s1, C is $s2, D is $s3.

```
A = B - C + A;
```

Consider this pseudo-instruction:

blt $t0, 10, Loop

(1 pts) Why did the MIPS designers make this a pseudo-instruction?

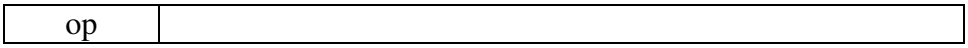
(2 pts) Show how to translate the above instruction into one or more real MIPS instruction.

(8 pts) For each of the following, circle whether it is a valid or invalid raw MIPS instruction (count pseudo-instructions as invalid). If invalid, briefly state why (must provide correct reason for credit if invalid).

`add $s0, $zero, $zero`	VALID	INVALID because:
`lw $s0, 0($s0)`	VALID	INVALID because:
`beq $t0, 17, L3`	VALID	INVALID because:
`sw $s2, $t1($t0)`	VALID	INVALID because:

(3 pts) Suppose register $s0 holds the base address of array B (an array of 32-bit integers). Show the MIPS instruction(s) to read element B[4] into register $s2.

(6 pts) Below are pictures of three different types of MIPS instructions. They are given only roughly to scale and some fields are not labeled. Below each picture, give the name of that type of instruction **AND** give an example of a specific instruction of that type (just the name of the instruction is fine, don't need to write a complete statement with registers etc.)



Instruction type? Example instruction?

Instruction type? Example instruction?

Instruction type? Example instruction?

(2 pts) In the pictures above, how many bits are in the "rs" field? **Why is this so?**

(5 pts) Suppose we already have this k-Map. Minimize the function and show final <u>sum of products</u> result.

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	0	0
$\overline{A}B$	0	1	0	1
AB	0	1	0	0
$A\overline{B}$	1	1	0	1

(3 pts) Draw the circuit for the following equation:

$z = (\overline{A} + B)(\overline{B} + C)$

(3 pts) Draw a picture (external view) of a multiplexor with <u>8 data inputs.</u>

(4 pts) Simplify the following equations:

$\overline{(AB)} \bullet (A + B + \overline{B}) =$

$(A + 0)(B \bullet 1)(C + \overline{C})(D + 1) =$

INSTRUCTIONS: Where needed on this page, assume the following variable/register mappings:

```
f - $s0
g - $s1
h - $s2
i - $s3
```

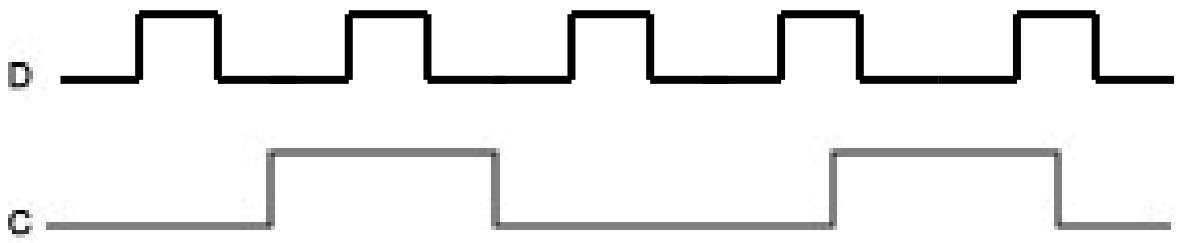
(6 pts) Translate the following C++ code into MIPS assembly.

```
for (i = 5; i < g; i++) {
     f = f + h;
}
```

(10 pts) Translate the following C++ code into MIPS assembly. Be sure to obey register conventions.

```
int function1 (int a, int b) {
     return function2(7, a) + b;
}
```

(8 pts) You are given a <u>rising edge</u> triggered D flip-flop. Complete the timing diagrams shown below for the output Q-FlipFlop (the output Q is shown initially low).



Q-FlipFlop

(4 pts) Circle the correct option for each question

Register files normally store each bit with a…	Latch	Flip-Flop	XOR
Which kind of logic remembers previous inputs?	Sequential	Combinational	Relational