

CSE 31

Sample Final

Time : 180 minutes

Name:

Problem	Points	Max Points
1		50
2		90
3		60
4		30
5		70
Total		300

1: [50 pts] Programming Languages

- a) Explain why : Mark & sweep is better than reference counting

- b) Explain why : Reference counting is better than copying

- c) Explain why : Copying is better than mark & sweep

- d) Explain why : The buddy scheme is better than the slab allocator

- e) Explain why : The slab allocator is better than the plain K&R free list

f) One reason why : Compiling is better than interpreting

g) One reason why : Interpreting is better than compiling

h) One reason why : Dynamic linking is better than Static linking

i) One reason why : Static linking is better than Dynamic linking because

j) One reason why : C is better than Java because

2: [90 pts] Number Representations

- a) [5 pts] A Binary Coded Decimal (BCD) uses a dedicated **nibble (a nibble contains 4 bits)** for each decimal digit, so a byte could represent all the numbers from 00-99. We will use our standard MIPS 32-bit word to encode a BCD. What is the ratio (to one significant figure, in decimal) of overall bit patterns to the ones that encode a valid BCD? (E.g., With a single decimal digit, it'd be $16/10 \approx 2$.) Show your work. Your answer should not be an expression, it should be a decimal number rounded to 1 significant figure.
- b) [5 pts] Suppose we have a very small 4 pixel × 8 pixel grayscale video display where each pixel can independently be set to one of 4 shades of gray. How many unique images can possibly be displayed? Leave your answer in shortened format if you desire (e.g., 64K images, 8M images, etc).
- c) [10 pts] If we were to try to compare two floats using our MIPS signed integer compare **slt** (set less than), when would we get an incorrect answer (i.e., describe in English the set of all possible inputs that generate incorrect answers)? Assume neither encodes a NaN or ± 0 .
- d) [10 pts] Convert 12.8125 into single precision floating point Binary representation using IEEE 754 standard. Write your answer in both Binary and Hex formats. Show your work for partial credit.

e) [10 pts] Convert the following single precision floating point Hex representation into decimal number using IEEE 754 standard. Show your work for partial credit:
0x41AA0000.

f) [50 pts] Put the corresponding letters for each 32-bit value in order from least to greatest. [Hint: the question isn't asking you to write down what each one is, it only asks for the relative order!]

- A. 0xF0000000 (IEEE float)
- B. 0xF0000000 (2's complement)
- C. 0xF0000000 (sign-magnitude)
- D. 0xFFFFFFFF (2's complement)
- E. 0xFFFFFFFF (1's complement)
- F. 0xF1000000 (IEEE float)
- G. 0x70000000 (IEEE float)
- H. 0x7FFFFFFF (2's complement)
- I. 0x80000010 (IEEE float)
- J. 0x80000010 (unsigned)

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____

3: [60 pts] C/MIPS

a) [30 pts] Given the MIPS code below, write the equivalent C function below in the structure we've provided. Feel free to add comments to help your disassembly. To aid readability, you must use the variable names from our comments below in your C solution where appropriate.

```
foo:
addiu $sp, $sp, -12

sw $a0, 0($sp)      # src
sw $a1, 4($sp)      # size
sw $ra, 8($sp)
move $a0, $a1       #
addiu $a0, $a0, 1   #
jal malloc          #
move $t0, $v0       # dest
lw $t1, 0($sp)      # src
lw $t2, 4($sp)
addu $t2, $t2, $t1  # end
foo_loop:
beq $t2, $t1, foo_end #
lbu $t4, 0($t1)      #
ori $t4, $t4, 0x20    #
sb $t4, 0($t0)        #
addiu $t0, $t0, 1     #
addiu $t1, $t1, 1     #
j      foo_loop
foo_end:
sb $0, 0($t1)        #
lw $ra, 8($sp)        #
addiu $sp, $sp, 12
jr $ra
```

```
_____foo (_____src, _____size) {

    for(_____;_____;_____) {

    }

}
```

b) [10 pts] If `src` contained letters, what is a more appropriate name for the subroutine `foo`? (i.e., what would “`jal foo`” do, from the point of view of the caller?)
Hint: use the MIPS reference sheet

c) [10 pts] What if we called `foo` from `printf` as so: `printf("...format string...", foo(source, size))`. Why is this bad form? Hint: think about what would happen if this were done many times.

d) [10 pts] Let's say we removed the “`sb $0, 0($t1)`” instruction and then made the same call to `foo` from `printf` as in question (e) above: What are all the things that could happen?

4: [30 pts] Digital System

Given the truth table below:

A	B	C	O
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

a) [15 pts] Provide a sum-of-products expression for Output O as a function of A, B and C.

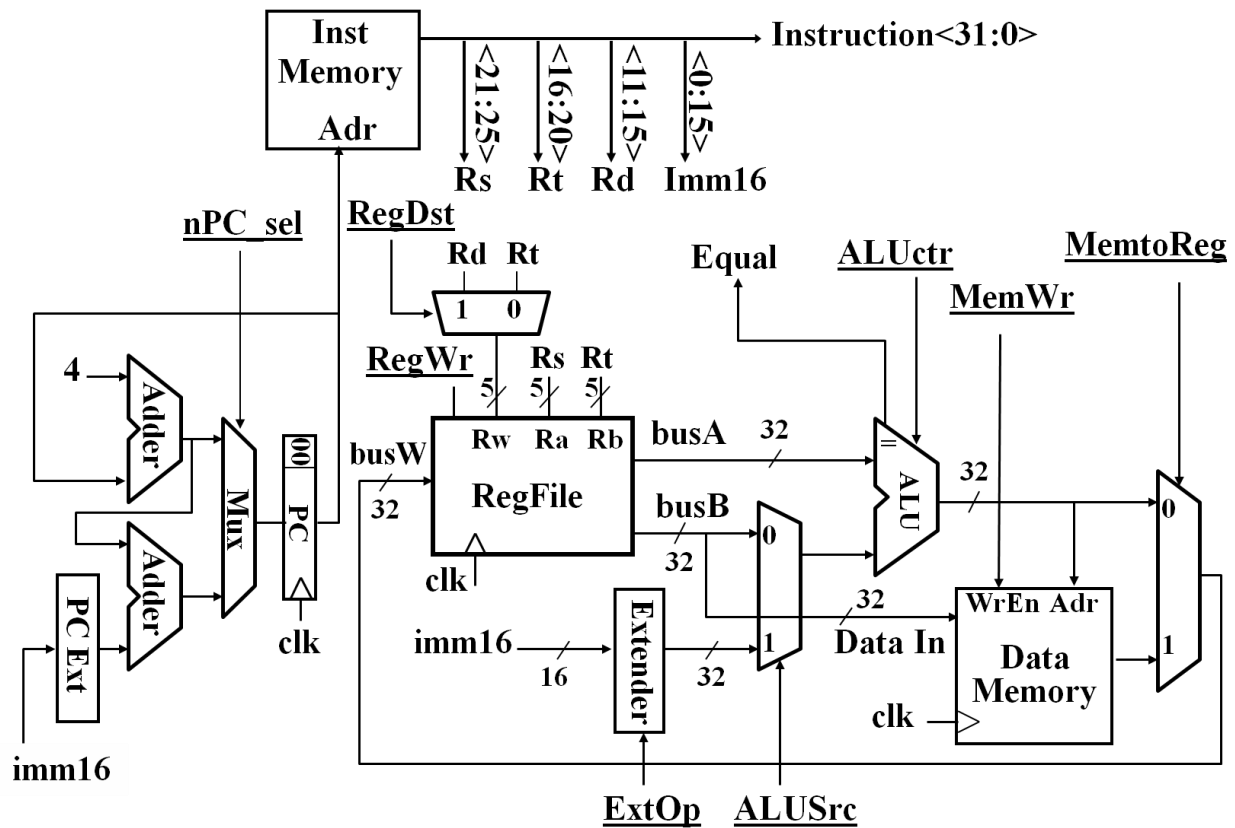
b) [15 pts] Draw a circuit diagram with logic gates for this system:

5: [70 pts] Cache / DataPath

a) [30 pts] Fill in the table

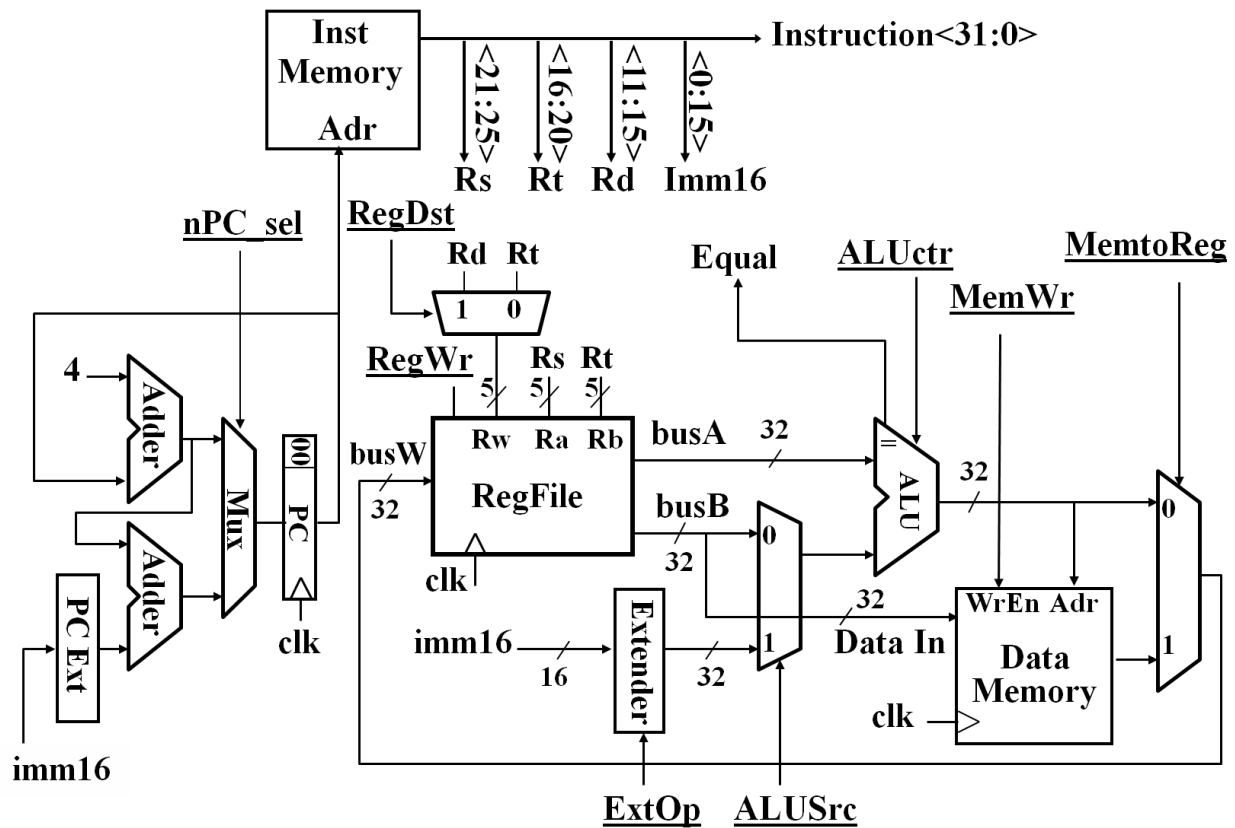
Address Bits	Cache Size	Block size	Associativity	Tag Bits	Index Bits	Offset Bits	Bits per row
16	4KB	4B	1				
16	16KB	8B	2				
32	8KB	8B	Full				
32	32KB	16B	4				
32	64KB			16	12	4	146
32	512KB		8		5		
64		64B	2	43	15		
64	2048KB		4				559

b) [20 pts] Trace through the paths the execution of LW instruction in this single cycle datapath design. Fill in the appropriate control signal values in the table. If there are any modifications needs to make it work then draw them in the diagram and create appropriate control values.



nPc_sel	ExtOp	ALUSrc	ALUctr	MemWr	MemtoReg	RegDst	RegWr

c) [20 pts] Trace through the paths the execution of J instruction in this single cycle datapath design. Fill in the appropriate control signal values in the table. If there are any modifications needs to make it work then draw them in the diagram and create appropriate control values.



nPc_sel	ExtOp	ALUSrc	ALUctr	MemWr	MemtoReg	RegDst	RegWr