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

Because it can handle circular structures

b) Explain why: Reference counting is better than copying

Because it can use all the memory. Copying only uses half.

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

Because it can recover all the free blocks including dis-continuous ones which get reabsorbed into bigger blocks.

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

Because it can split slabs of a bigger sizes when there are not enough slabs of smaller sizes.

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

Because it allows small size requests to be handled quickly.

f) One reason why: Compiling is better than interpreting

Because it creates faster executables and hide source code from the end-users.

g) One reason why: Interpreting is better than compiling

Because it allows portable code that runs on any machine and provides a fast process to go from coding to executing.

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

Because it leads to smaller executable with ability to incorporate updates easily.

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

It provides faster runtime and avoids having to link once executed. Users will not get missing DLL errors.

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

It creates faster executables.

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 ≈ 2.) Show your work. Your answer should not be an expression, it should be a decimal number rounded to 1 significant figure.

For MIPS 32-bit word, it can represent 2^{32} values
For 32-bit BCD, there are 8 nibbles (a nibble is 4 bits). It can represent 10^8 values
Ratio = $2^{32} / 10^8 \approx 40$

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).

Each image has 4x8 = 32 pixels, and each pixel can have one of 4 shades.
Therefore, the total number of images = $4^{32} = 2^{64}$

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.

The compare will get an incorrect answer when both operands are negative numbers. Since signed integers are represented as 2's complement and the exponent of a float is represented as biased, their increment of bits are in different directions.

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.

0100 0001 0100 1101 0000 0000 0000 0000 or 0x414d0000

Steps:
$12_{10} = 1100_2$ and $0.8125_{10} = 1101_2$
$1100.1101 = 1.1001101 \times 2^3$
Exponent = $127_{10} + 3_{10} = 130_{10}$ or $1000\ 0010_2$
Mantissa = 100 1101 0000 0000 0000 0000

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.

21.25

Steps:

0100 0001 1010 1010 0000 0000 0000 0000

sign = 0

exp = 10000011_2 - 127_{10} = 131_{10} - 127_{10} = 4_{10}

mantissa = $010\ 1010\ 0000\ 0000\ 0000\ 0000_2$

$1.010101 \times 2^4 = 10101.01 = 2^4 + 2^2 + 2^0 + 2^{-2} = 16 + 4 + 1 + 0.25 = 21.25$

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. F
2. A
3. C
4. B
5. D
6. I
7. E
8. H
9. J
10. G

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
```

```
char * foo (char *src, size_t size) {

     char *dest, *d, *end;

     dest = (char *) malloc ((size + 1) * sizeof(char));

     for(d = dest, end = src + size; d != end; src++)

          *d = *src | 0x20;

     *d = 0;

     return dest;
}
```

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

strNLowerCaseCopy

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.

There will be a memory leak.

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 (c) above: What are all the things that could happen?

Three possible outcomes:
- Segmentation fault.
- It prints output of foo correctly.
- It prints output of foo followed by some garbage.

4: [30 pts] Digital System

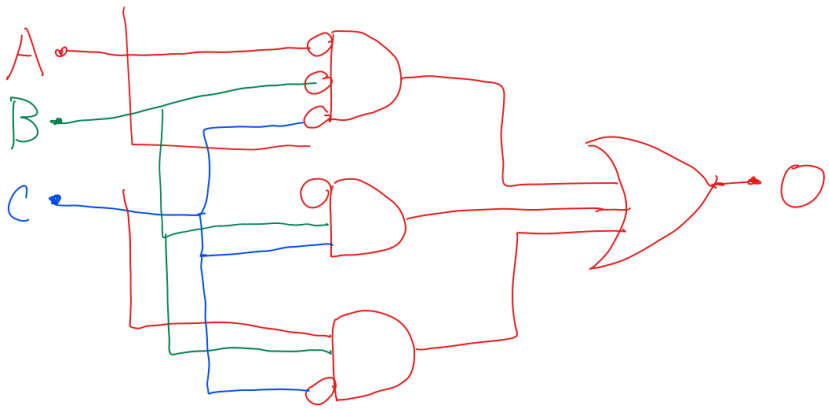
Given the truth table below:

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.

$$O = \bar{A}\bar{B}\bar{C} + \bar{A}BC + AB\bar{C}$$

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



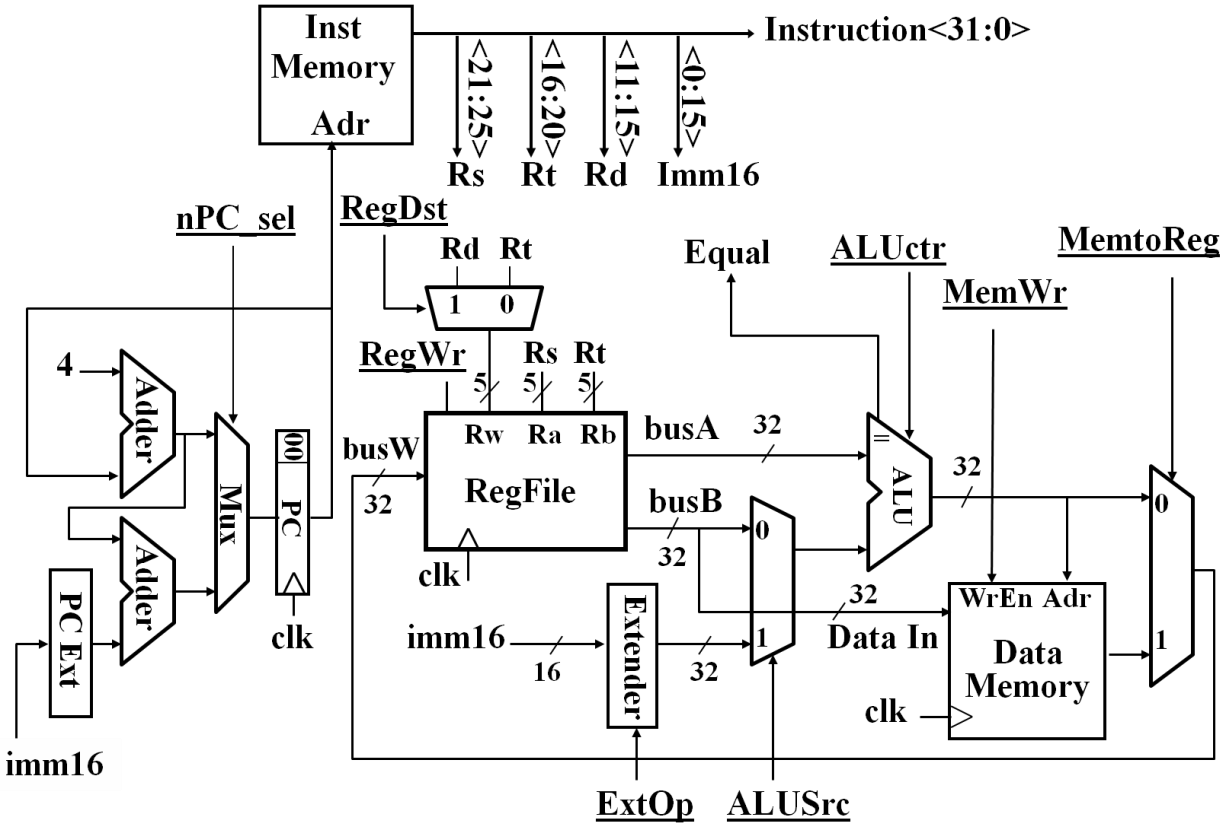
5: [70 pts] Cache / DataPath

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	4	10	2	38
16	16KB	8B	2	3	10	3	69
32	8KB	8B	Full	29	0	3	95
32	32KB	16B	4	19	9	4	149
32	64KB	16B	1	16	12	4	146
32	512KB	2KB	8	16	5	11	16402
64	4096KB	64B	2	43	15	6	557
64	2048KB	64B	4	45	13	6	559

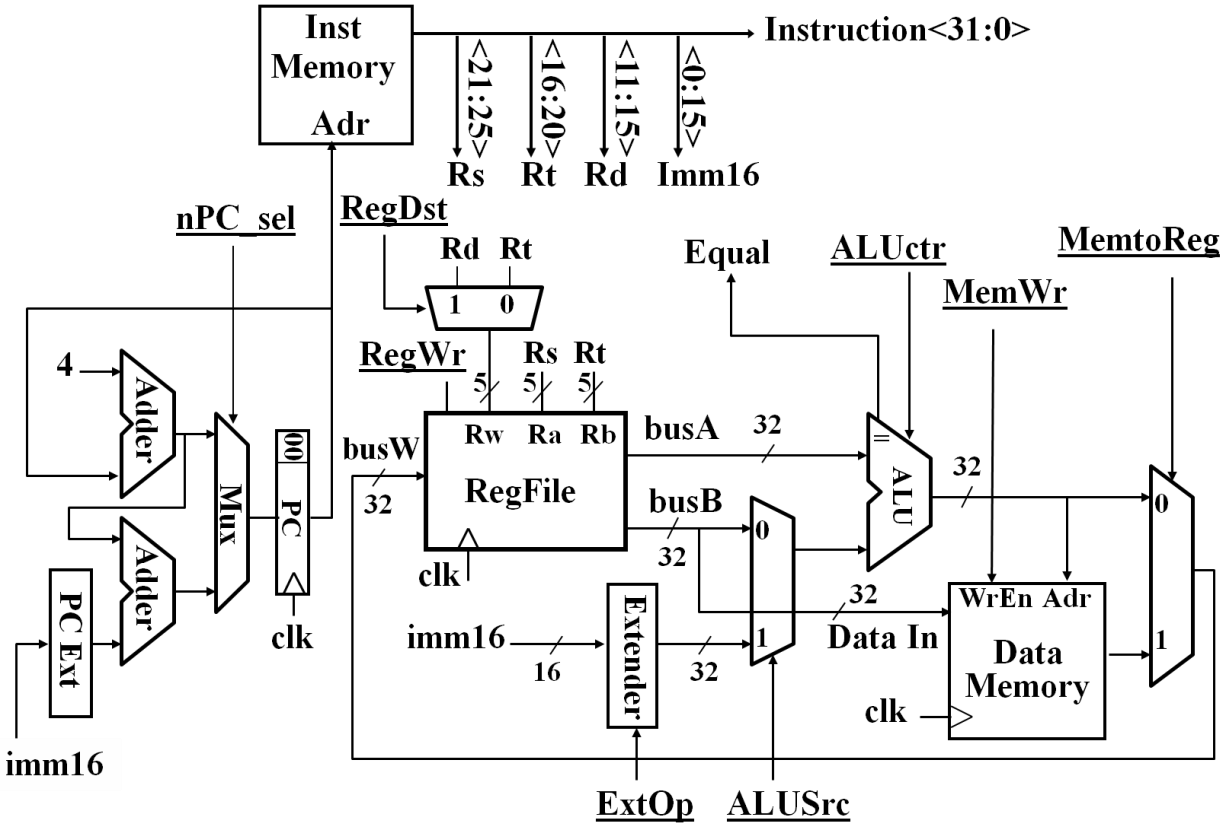
Bits per row = bytes * 8 bits + Tag bits + valid bit + dirty bit
4 * 8 + 4 + 1 + 1 = 38
8 * 8 + 3 + 1 + 1 = 69
8 * 8 + 29 + 1 + 1 = 95
16 * 8 + 19 + 1 + 1 = 149
2048 * 8 + 16 + 1 + 1 = 16402
64 * 8 + 43 + 1 + 1 = 557

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
0	1	1	ADD	0	1	0	1

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
j	x	x	x	0	x	x	0