COMP 230: Computer Architecture and Organization

Exam 1 (Practice)

Instructions:

- **This practice exam is meant to give you the flavor of questions that will be asked on the exam. Do not expect the real exam to be the same questions with only the numbers or MIPS instructions changed.**
- **This practice exam is slightly longer than the real exam. However, the real exam will contain questions of similar difficulty, with the exception of the last question.**
- Your real exam will be open book and notes. However, you are not allowed to use laptops, cell phones, calculators, or other electronics.
- If you do not show your work, do not expect partial credit for incorrect answers.
- If you believe a problem is incorrectly or incompletely specified, make a reasonable assumption and solve the problem. The assumption should not result in a trivial solution.
- In all cases, clearly state any assumptions you make in your answers.

Name	

Question	**Points Possible**	**Grade**
1	10	
2	20	
3	10	
4	20	
5	20	
6	20	
Total	100	

1. Your employer has tasked you with comparing two processors and picking the best one. Unfortunately, the manufacturers don't want to give out all the information. Here's what you know:

	Processor A	Processor B
Clock Rate	5GHz	?
Cycle Time	?	.5 nanoseconds

(a) What is the clock rate for processor B?

$$\text{Clock rate} = \frac{1}{\text{cycle time}} = \frac{1\,\text{cycle}}{0.5\,\text{ns}} = \frac{1\,\text{cycle}}{0.5 \times 10^{-9}\,\text{s}} = 2 \times 10^{-1}\,\text{cycle/s} = \boxed{2\,\text{GHz}}$$

(b) What is the cycle time for processor A?

$$\text{Cycle time} = \frac{1}{\text{clock rate}} = \frac{1\,\text{s}}{5 \times 10^{9}\,\text{cycle}} = 0.2 \times 10^{-9}\,\text{s/cycle} = \boxed{0.2\,\text{ns}}$$

(c) What other information would you need to decide which processor is better?

You need to know the CPI. Even then, in real life you will care about execution time of some programs but not others. So actually you also need to know the number of and types of instructions in the program(s) you care about.

2. The program your company wants to execute on the processor needs only three types of instructions: 20% are integer math operations, 50% are conditionals, and 30% are memory accesses. Your program has $100,000,000$ total instructions. Your processor has a clock rate of 2 GHz and you know the CPI for each type of instruction:

Instruction Type	CPI for current processor
Integer Math	5.0
Conditionals	2.0
Memory Access	10.0

(a) What is the average CPI for the program?

$$\text{CPI} = 0.2(5) + 0.5(2) + 0.3(10) = 1 + 1 + 3 = \boxed{5\,\text{cycle/s}}$$

(b) What is the CPU execution time?

$$\begin{aligned}\text{Total Execution Time} &= \frac{100\times 10^6\,\text{instruction}}{\text{program}} \times \frac{5\,\text{cycle}}{\text{instruction}} \times \frac{1\,\text{s}}{2\times 10^9\,\text{cycle}} \\ &= \frac{250\times 10^6}{1\times 10^9}\text{s} = 250\times 10^{-3}\,\text{s} = \boxed{250\,\text{ms}}\end{aligned}$$

(c) How much time is spent on integer math instructions?

$$\begin{aligned}\text{Total Execution Time} &= \frac{0.2\times 100\times 10^6\,\text{instruction}}{\text{program}} \times \frac{5\,\text{cycle}}{\text{instruction}} \times \frac{1\,\text{s}}{2\times 10^9\,\text{cycle}} \\ &= 50\times 10^{-3}\,\text{s} = \boxed{50\,\text{ms}}\end{aligned}$$

(d) The CPU manufacturer of your processor now claims they have a new ALU which improves the CPI for integer arithmetic by a factor of 2. What is the improved execution time for processor B?

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}} = \frac{50\,\text{ms}}{2} + 200\,\text{ms} = \boxed{225\,\text{ms}}$$

3. Convert the following to binary and add, then convert the result into hexadecimal: $-160 + 90$.

$$\begin{aligned} 160 = 0000\ 1010\ 0000 \implies -160 &= 1111\ 0110\ 0000 \\ 90 &= \underline{0000\ 0101\ 1010} \\ & \\ &\ \ 1111\ 1011\ 1010 = -70 \\ &= \boxed{\text{fba}_{16}} \end{aligned}$$

4. Consider the hexadecimal instruction `8d10 000c`.

(a) Convert the instruction into MIPS assembly.

```
lw $s0, 12($t0)
```

(b) Convert the instruction into (binary) machine code.

8	d	1	0	0	0	0	c
1000	1101	0001	0000	0000	0000	0000	1100

`op` is $100011 = 35 = 23_{16}$, which is `lw`, so this is an I-format instruction.

op	rs	rt	imm
100011	01000	10000	0000 0000 0000 1100
23_{16}	8 = `$t0`	16 = `$s0`	12

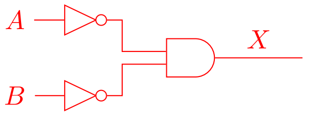
5. Consider the following logic functions:

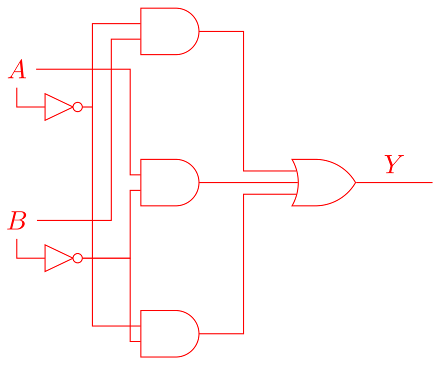
$$\begin{aligned} X &= (\overline{A+B}) \\ Y &= (\overline{A} \cdot B) + (A \cdot \overline{B}) + (\overline{A} \cdot \overline{B}) \end{aligned}$$

(a) Convert X to sum of products form using one of DeMorgan's laws.

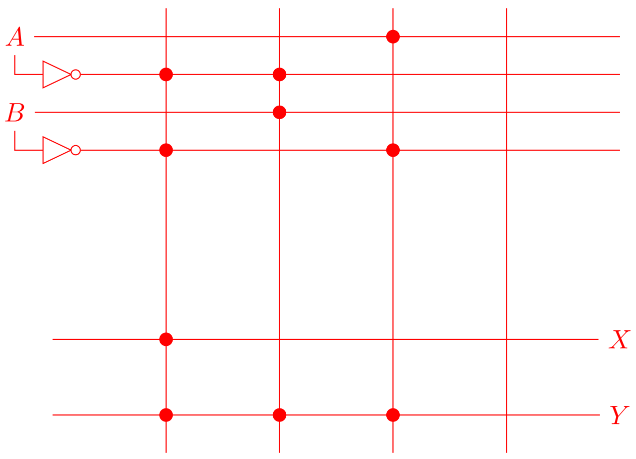
$X = (\overline{A} \cdot \overline{B})$

(b) Draw separate wire (gate) diagrams for each.





(c) Draw a single PLA for these functions. You may use either of the two PLA notations we have learned.



6. Consider the following C code that operates on an array of (32 bit) integers:

```
int calc(int size, int data[]) {
  for (int i{0}; i < size; i++) {
    data[i] = foo(data[i]) + (i / 16);
  }
  return data[size - 1];
}
```

Write the code in MIPS assembly.

- The procedure `foo` is just a helper procedure — don't worry about exactly what it does.
- You are NOT allowed to use any `mul` instructions.
- **This question is slightly harder than what will be asked on the exam, since you will be limited by time.**

Note: there are many acceptable answers for this question.

```
calc:      addi  $sp,  $sp,    -20
           sw    $ra,  0($sp)           # Since this is a non-leaf procedure
           sw    $s0,  4($sp)           # Use $s0 for i
           sw    $s1,  8($sp)           # Use $s1 for size
           sw    $s2,  12($sp)          # Use $s2 for data
           sw    $s3,  16($sp)          # Use $s3 for current data[i] address
           add   $s2,  $a1,    $zero    # Save data
           add   $s1,  $a0,    $zero    # Save size
           add   $s0,  $zero,  $zero    # int i{0}
loop:      slt   $t0,  $s0,    $s1      # Check i < size
           beq   $t0,  $zero,  done
           sll   $s3,  $s0,    2        # Distance from index i to base address
           add   $s3,  $s3,    $s2      # Compute address of data[i]
           lw    $a0,  0($s3)           # Load data[i] to prepare for foo()
           jal   foo                    # call foo(data[i])
           srl   $t0,  $s0,    4        # i / 16
           add   $v0,  $v0,    $t0      # foo(data[i]) + (i/16)
           sw    $v0,  0($s3)           # Store above result into data[i]
           addi  $s0,  $s0     1        # i++
           j     loop
done:      lw    $v0,  0($s3)           # Trick: s3 still holds the address of data[size-1]!
epilogue:  lw    $s3,  16($sp)
           lw    $s2,  12($sp)
           lw    $s1,  8($sp)
           lw    $s0,  4($sp)
           lw    $ra,  0($sp)
           addi  $sp,  $sp,    20
           jr    $ra
```