ITSC 3181 Introduction to Computer Architecture, Spring 2023

Homework #3, Due on 03/07

Covered topics: Numbers, CPU performance, C to assembly, and logic design

Question	1	2	3	4	5	6	7	Total
Total Points	15	15	15	15	15	15	30	120
Grade								

5% of the grade of this homework will be added to your final grade.

Submission:

1. **Only electronic submissions on Canvas are accepted. All your solutions should be included in a SINGLE PDF file. Include your full name in the PDF file.**
2. **Number your solutions in the same way and in the same order as the questions are numbered in this document.**
3. **You must show the necessary steps when you solve each question. 0 point will be given to the question if the answer only includes the final answer with no necessary steps.**
4. **Scanned copy of handwritten work will be accepted and graded only if it is written clearly and readable.**

1. Convert the following decimal numbers to 6-bit 2's complement binary numbers and perform additions. Indicate whether or not the sum overflows a 6-bit result.
   a. 16 + 9
   b. 27 + 31
   c. -4 + 19
   d. 3 + -32
   e. -16 + -9
   f. -27 + -32

2. Using only **add**, **addi**, **sub**, **bne**, **beq** (branch if equal) and **bgt** (branch if greater than) to implement the R-type instruction for multiplication "mul rd, rs1, rs2" for unsigned integer which could positive or zero, and assume no overflow. You can use temp register t0, … t6.

3. Suppose you must write a sequence of instructions that scans an array of 3 integers to find its minimum value (which is stored in register $s2). You are considering the following two implementations.

Implementation #1

```
        li $s0,0
        li $s1,12
        li $s2,1000
loop:   beq $s0,$s1,exit
        lw $t0,vals($s0)
        blt $t0,$s2,gotone
        j skip
gotone: or $s2,$0,$t0
skip:   addi $s0,$s0,4
        j loop
exit:
```

Implementation #2

```
        li $s0,0
        li $s1,12
        li $s2,1000
loop:   lw $t0,vals($s0)
        bge $t0,$s2,skip
        or $s2,$0,$t0
skip:   addi $s0,$s0,4
        blt $s0,$s1,loop
```

The number of cycles required for each instruction type are shown below:

Instruction type	Cycles	#instrs of Impl #1	#instrs of Impl #2
R-type and ADDI	2		
Branch and Jump	5		
Load and Store	9		
Load Immediate	1		
Total Cycles:	X		

Assume the values stored in the vals array is [12,10,8]. Determine which implementation is faster and by how much. In this question, you need to simulate the execution manually and count how many instructions are executed for each of the instruction type. And then calculate the total number of cycles used by each implementation and compare which uses less (faster) and the ratio of cycles.

4. The following C program fill in an array C with the larger elements from another two arrays A and B. Convert the code to its RISC-V assembly code. For reference to array elements A[i], you can use A(i) as its base+offset representation in the LW/SW instruction, e.g. to load an integer from A[i], you can use LW x5, A(i); Use register t0 and t1 for storing i and N respectively.

```
int i;
for (i=0; i < N; i++) {
    if (A[i] >= B[i])
       C[i] = A[i];
     else
       C[i] = B[i];
}
```

5. We use a processor with CPIs for the following classes of instructions to execute the following C-code. 1) Count how many instructions are to be executed for each of the class, and 2) calculate the total number of cycles for each class and overall total cycles for all classes, and 3) calculate the percentage of total cycles by each class with respect to the overall total cycles. Addition used for calculating memory effective address for load and store instructions should NOT be considered. But loop count increment is considered as addition.

Instruction class	CPI
add	1
mul	20
load/store	2
branch	2

```
int N = 1000,000;
int A[N], B[N], i;
for (i=0; i < N; i++) {
    A[i] += 3.14 * B[i]
}
```

Instruction class	CPI	# instructions x10^6	Total Cycles x10^6	Cycles Percentage
add	1			
mul	20			
load/store	2			
branch	2			

Overall Total cycles =

6. For the same processor as problem 4, answer the same questions for the following C code.

```
int N = 1000;
int A[N][N], B[N]
int i, j;
for (i=0; i < N; i++) {
   int temp = 0;
   for (j=0; j < N; j++) {
     temp += A[i][j] * B[j]
    }
    C[i] = temp;
}
```

Instruction class	CPI	# instructions x10^6	Total Cycles x10^6	Cycles Percentage
add	1			
mul	20			
load/store	2			
branch	2			

Overall Total cycles =

7. 1) Write a Boolean equation in sum-of-products canonical form for each of the following five truth tables. 2) Then minimize each of the Boolean equation, and 3) sketch a combinational circuit that implement the minimized Boolean equation using only NOT, AND and OR gates.

(a)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(b)

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(c)

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

(d)

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

(e)

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

The following exercise, together with provided answers is for your study. They are the similar question to question 7. 1) Write a Boolean equation in sum-of-products canonical form for each of the following truth tables. 2) Then minimize each of the Boolean equations, and 3) sketch a combinational circuit that implements each minimized Boolean equations using only NOT, AND and OR gates.

(a)

A	B	Y
0	0	1
0	1	0
1	0	1
1	1	1

(b)

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(c)

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

(d)

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

1) Boolean equation in SoP forms of the above truth tables:

(a) $Y = \bar{A}\bar{B} + A\bar{B} + AB$

(b) $Y = \bar{A}\bar{B}\bar{C} + ABC$

(c) $Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + ABC$

(d)

$$Y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + ABC\bar{D}$$

2) Minimized Boolean equation:

(a) $Y = A + \bar{B}$

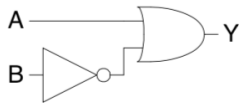
(b) $Y = \bar{A}\bar{B}\bar{C} + ABC$

(c) $Y = \bar{A}\bar{C} + A\bar{B} + AC$

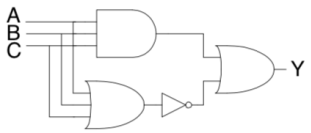
(d) $Y = \bar{A}\bar{B} + \bar{B}\bar{D} + AC\bar{D}$

3) Circuit implementation using AND, NOT and OR:

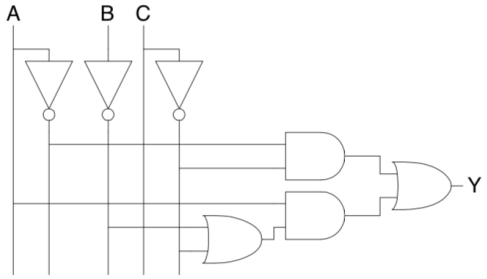
(a)



(b)



(c)



(d)