International Institute of Information Technology Hyderabad

Computer Systems Organization [Sections B1 and B2]

Assignment 2
Architectural Exploration
Deadline: July 07, 2021 (Wednesday), 23:55 PM
Total Marks: 100

1 Instructions:

- There are **4 NEW INSTRUCTIONS** mentioned in this assignment where each one of you will be working on one instruction which will be (RollNumber % 4 + 1). For example if your Roll Number is 2018111016 you will be working on instruction 4 (since 2018111016 % 4 + 1 = 1).
- This is a theoretical based assignment which needs to be **TYPED** in Google Docs or Latex. (**Handwritten is strictly not accepted**). In case you are using any hand-drawn figures for some data, then you need to click them as an image and put it in the TYPED file appropriately.
- You will need to submit a PDF file named RollNumber.pdf
- **MOST IMPORTANT NOTE: PENALTY for PLAGIARISM will be more SEVERE than Assignment1 which might lead to an F in the COURSE, so sincere advice to not involve in any such actions.**

2 Overview:

The objective of this assignment is to assess your knowledge of **Chapter 4** from the book Computer Systems-A Programmers Perspective, Bryant and O'Hallaron. In the course of this assignment you will explore micro-architectural design ideas.

3 Problems

You have to support **ONE** new type of instruction in the Y86 ISA. Following is the list of new instructions.
Note: Your assigned instruction will be (RollNumber % 4 + 1) as mentioned in above Instructions section.

1. **Instruction 1**
   [OPI] rA , D(rB) :- Execute operation OPI. The first operand is stored in register rA and the second in memory. The memory address is composed by adding the base address stored in rB with displacement value D. The result of the operation is stored in the memory address. Here OPI can be any one of the operations given in Figure 4.3 of chapter 4. The registers can be any of the registers from Figure 4.4 of chapter 4.
   Example - addl %eax, 100(%ebx).

2. **Instruction 2**
   p[OPI] :- Execute the operation OPI on the top two values of the stack and store the result back on the top of the stack. Here OPI can be any one of the operations given in Figure 4.3 of chapter 4. The registers can be any of the registers from Figure 4.4 of chapter 4.
   Example - paddl [exp: add the top two values on top of the stack and push the result back on the stack].

3. **Instruction 3**
   cxchng[XX] rA , rB :- Swap the values in rA and rB. The swap only happens if the condition [XX] is satisfied. Here [XX] can be any one of the jump conditions given in Figure 4.3 of chapter 4. The registers can be any of the registers from Figure 4.4 of chapter 4.
   Example - cxchngge %eax, %ebx [Explanation - Exchange %eax with %ebx only of %ebx ¿= %eax].

4. **Instruction 4**
   ccall[XX] Dest :- Execute the call instruction if the condition [XX] is satisfied. Here [XX] can be any one of the conditions given in Figure 4.3 of chapter 4.
   Example - ccallne dest [explanation - execute the instruction in Dest if the previous operation result in a true value for the not equal to condition]

4 Tasks

Now once you are assigned with your instruction you need to perform below 7 tasks on the instruction assigned to you, for which you need to type your answer in the doc with the task number as sub heading. Make sure you mention all your assumptions (if any) and give a detailed information about your answer.
NOTE: Make sure you follow the similar structure used in the referred sections of textbook to write your answer.
Need to follow the sequential order of tasks as the later tasks depend on previous ones.

(a) **Task 1**
Come up with an instruction encoding for your new instruction. Recall that Y86 instructions can be maximum 6 bytes long. Refer to section 4.1 chapter 4 of the book for this task. [5]

(b) **Task 2**
Write the Y86 instructions (Comments are mandatory) for **ONE** of the below problems. Based on the index of your instruction in the list above you have to choose the same index for the problem below (i.e (RollNumber % 4 + 1)). For example if you are assigned Instruction 1, then you have to choose Problem 1 only for implementation. Refer [15]

   i. **Problem 1**
   Given an array of 8 unsigned integers in memory. Increment the values of 4 integers within the array by a specific amount. The 4 index values are stored in a separate memory array. The increment value is stored in a register. Along with other instructions you have to use **[OPI] rA , D(rB)** instruction in this problem.

   ii. **Problem 2**
   Find the sum of an array of 8 unsigned integers in memory. Along with other instructions you have to use **p[OPI]** instruction in this problem.

   iii. **Problem 3**
   Given an array of 8 unsigned integers. Swap the leftmost 4 values with the right most 4 values in the array. The swap will only happen if the element on the right is greater than the element on the left. For example element 0 will be swapped with element 7 if e[7] > e[0], element 1 with element 6 if e[6] > e[1] and so on. Along with other instructions you have to use **cxchng[XX] rA , rB** instruction in this problem.

   iv. **Problem 4**
   Given an array of 8 unsigned integers. Swap the leftmost 4 values with the right most 4 values in the array. You will have to use a **swap function** written by you. The swap function will only be called if the element on the right is greater than the element on the left. For example element 0 will be swapped with element 7 if e[7] > e[0], element 1 with element 6 if e[6] > e[1] and so on. Along with other instructions you have to use **ccall[XX] Dest** instruction in this problem.

(c) **Task 3**
Write the memory dump of the instructions written by you for the problem in the previous task (i.e Task 2). Refer to Figure 4.8 (left part) of chapter 4 from the book. Assume the starting address is 0x0000. [15]

(d) **Task 4**
After completing task 3 you will get an address where the instruction assigned to you is located. If you have used more than one instance of your new instruction in the program, then pick up the instance with the lowest address. For this instance you have to create the Fetch, Decode, Execute, Memory, Write Back, PC Update Table. This table will capture how your instruction gets implemented across these stages on the Y86 architecture. There are a lot of examples of this table in the book for the existing instructions. Refer to Figure 4.18, 4.19, 4.20, 4.21 of chapter 4 in the book.
Note: You only have to write the table for one instance of your new instruction. [15]

(e) **Task 5**
Starting from the first instruction in your Y86 code of your program in Task 2, trace 20 cycles of execution on the sequential Y86 architecture. For each cycle, report the value of the program counter, condition code registers, the general purpose registers, stack pointer (if it is used), and any modified addresses in memory. You will report this data in a table. The first column will be the cycle ID [1,2,3 . . .] and the other columns will be for the state elements, one column per state element. Refer to section around figure 4.25 of chapter 4 in the book for this task. [20]

(f) **Task 6**
In the pipelined implementation of the Y86 architecture (section 4.4 - 4.5 of the book). Find the number of cycles required by your program to execute. Draw the pipelined cycle diagram for the first 20 instructions in your program. Refer to figure 4.4.3 of chapter 4 from the book. [20]

(g) **Task 7**
Come up with any new Y86 Instruction, that will require you to add new hardware block/control signals to the Y86 architecture. Briefly describe the instruction and how it requires a new hardware block. [10]

POP $(All the best!!!)