IIT JODHPUR

Minor Examination: EEL1530 Processor Design (OPEN-BOOK)

Guidelines (Total time: 120 minutes, Maximum Marks: 25):

- Please read the question paper very carefully. **NO clarification is required in any question. In case of any doubt, assume whatever you wish to and state that in your answer.**
- Usage of internet/any LLM tool is NOT allowed. Answer to-the-point ONLY in legible handwriting.

---

1. You are responsible for the design of a high performance system, $TharV1$. You decide to vectorize the instructions (as shown below) in this processor. LHS shows C code and RHS shows respective instructions.

```
for i := 1 to 64 do
  for j := 1 to 64 do
    Y[k,j] := a * X[i,j] + Y[k,j];
```

```
R1 := 0
R2 := 0
R3 := k*N

for i := 1 to 64 do
    LV      V2,X(R1)   ; load row of X (R1 is i*N, where N is row length)
    MULTSV  V3,a,V2    ; multiply each element of the row vector by scalar 'a'
    LV      V1,Y(R2)   ; load k'th row of Y, where R2 is k*N)
    ADDV    V4,V3,V1   ; add Y[i] and a*X[i]
    SV      V4,Y(R3)    ; store the vector into array Y

    R1 := R1+N
    R2 := R2+N
end do
```

   It is known to you that when a computation is run in vector mode, it is 20 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode as the **percentage of vectorization**. Find out the **percentage vectorization** that is needed to achieve one-half of the maximum speedup available from using vector mode in $TharV1$ processor? [4]

2. As an intern, you are tasked with the design of a new hardware-based branch predictor $BPstar$ in the $TharV2$ design of a startup incubated at IIT Jodhpur. This branch predictor essentially tracks the history of program counter values (each of 4 bytes) and the history of branch decisions taken at the respective addresses. It is known that if the last byte of the address is divisible by 3, branch history is always NOT TAKEN and if the address is divisible by 2, it is always TAKEN and in all other cases, it is always NOT TAKEN. Given that the misprediction penalty is 4 cycles (while the correct branch instruction execution takes 1 cycle) and total branch instructions is 1000, **assuming random branch outcomes**, compute the speedup with this predictor compared to the case where you have a fixed branch predictor which predicts alternately in this sequence: TAKEN, NOT TAKEN, TAKEN, NOT TAKEN, TAKEN, NOT TAKEN, TAKEN, NOT TAKEN and so on. Can you improve $BPstar$ in some way? [4+1]

3. You are working as an application developer for a software company $PicoSoft\ Technologies$. You have developed a special application that takes as input an image and then provides a caption and describes the image contents in three/four sentences. You are provided 2 options for executing your innovative Image-to-Text software/application: Processor A: 4 cores, 3.5 GHz clock speed and Processor B: 8 cores, 4.0 GHz clock speed. The program consists of following instruction distribution & CPIs for each class:

   - Arithmetic Instructions: 40%, CPI = 1.5
   - Memory Access Instructions: 30%, CPI = 2.0
   - Branch Instructions: 30%, CPI = 2.5

   The program has a parallelizable portion (P) of 70% and a sequential portion (S) of 30%. Devise a scheme to reduce percentage of branch instructions so that Processor A matches the performance of Processor B. You can assume that decreasing the percentage of branch instructions does not change the values of S and P portions. Also, assume that decreasing branches reduces the total number of instructions in the program. [Hint: Devising a scheme means quantify & justify the percentage of instruction reduction.] [5]

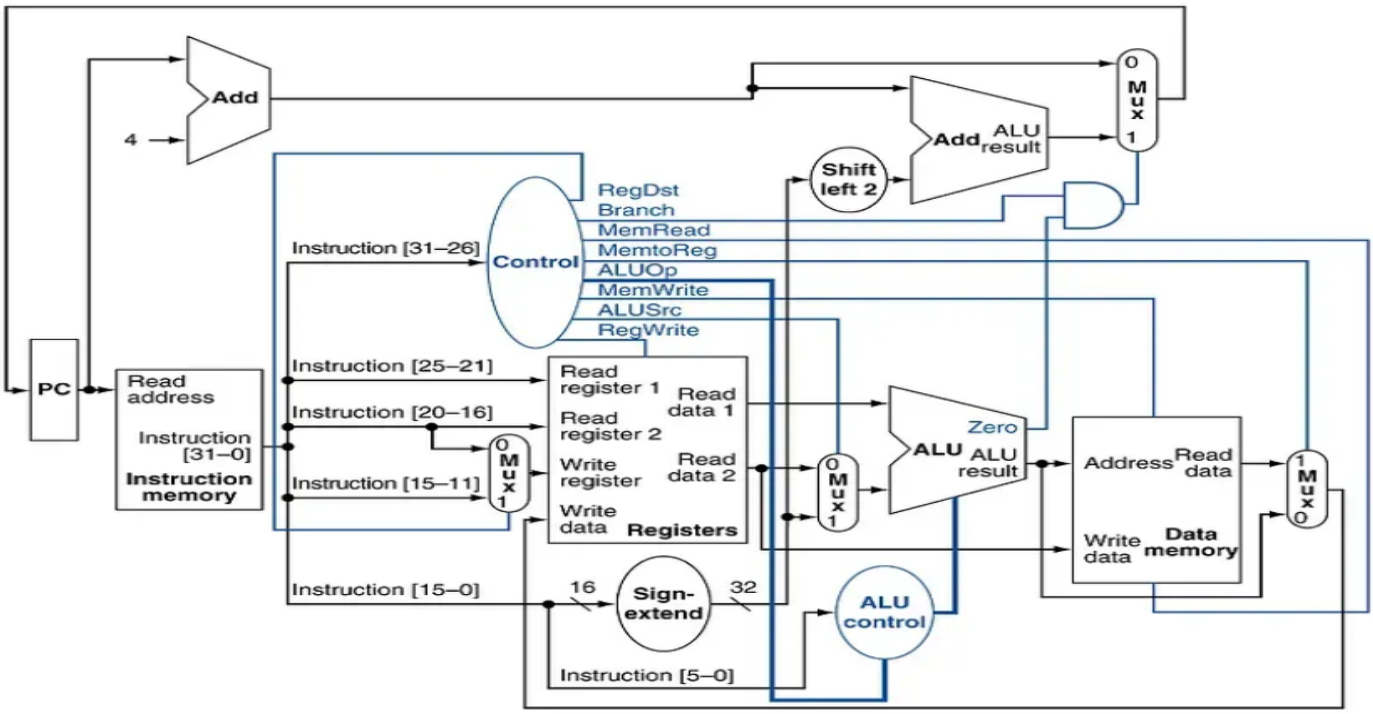
4. Suppose you are hired as a consultant to the local investing firm *MewarDollars Inc.* The application developed by this company primarily solves a system of linear equations for stock price optimization. In this software, matrix element manipulation (as shown below) is heavily used for FP data inputs.

```
for i := 1 to 100
  Y[i] := a*X[i] + Y[i];
```

```
foo:  LD     F2, 0(R1)        ; load X[i]
      MULTD  F4,F2,F0         ; multiply a*X[i]
      LD     F6, 0(R2)        ; load Y[i]
      ADDD   F6,F4,F6         ; add aX[i] + Y[i]
      SD     0(R2),F6         ; store Y[i]
      ADDI   R1,R1,8          ; increment X index
      ADDI   R2,R2,8          ; increment Y index
      SGTI   R3,R1,done       ; test if finished (R3 := (R1>done))
      BEQZ   R3,foo           ; loop if not finished.
```

LHS shows C code and RHS shows respective instructions. Assume integer operations issue and complete in one clock cycle. FP addition takes 2 cycles, FP multiplication takes 5 cycles, and FP division takes 19 cycles. Assume a static pipeline with separate, FP units, one for addition/subtraction, and one for multiplication/division. How many clock cycles does each iteration of the loop (shown above) take? Explain any way to optimize this design further? [3+2]

5. You are acting as a design manager in a local startup *BharatProc Private Limited.* One of your team members have recently switched to another company. Before the exit, he handed over the below diagram of single cycle datapath (designed by him) for a processor design based on open-source RISCV ISA.



You have few interns in your team to implement this design in RTL. They request you to explain this datapath in detail so that they can kick start the RTL coding. Therefore, **separately** draw the datapath for R-type and memory-type instructions (i.e., in total 2 diagrams are required) and briefly explain how the design of datapath is actually efficient and instructions get executed in a single cycle itself? [4+2]
