Computer Architecture (CS F342)
Semester-I, 2022-23
Midsem Examination
Department of Computer Science and Information Systems (CSIS)
BITS-Pilani, K K Birla Goa Campus, Goa, India.
IC: Dr Kanchan Manna

Date: Nov 03, 2022: 2:00 PM **Duration: 1:30 Hrs** **Total Marks: 90**

Instructions:

For the questions, no marks would be awarded if no reasoning is found in the answer script. You must write all answers of a question contiguously [not here and there within the answer script].

1. (a) Why do we have opcode field and function field in the instruction format for MIPS ISA instead of only opcode field? Justify your answer.
   (b) Why is it called the control unit's Truth Table when we consider the control unit's control generation for single-cycle datapath?
   (c) Which instructions will be affected, consider only those instructions discussed in the class, if left-shift by 2-bits operation, in Single-Cycled MIPS microprocessor, works as the left-shift by 1-bit? Explain such an effect.
   (d) The pipelined MIPS microprocessor is running the following program. Which registers are being written, and which registers are being read on the fifth cycle?

```
or $t2, $t4, $t5
sub $t0, $t1, $t2
sw $t5, 72($t0)
addi $t1, $t1, 50
lw $t3, 15($t1)
```

[**Marks: 2 + 2 + 5 + 2 = 11**]

2. (a) Consider MIPS microprocessor and provide the type and assembly language instruction for the following binary values stored in memory location starting from M0:
   M3 [0010 0000], M2 [0100 0000], M1 [0000 1000], M0 [0000 0001]
   (b) The MIPS register file should be expanded to 128 registers, and the instruction set should be expanded four times. What would be the effects of this on the size of bit fields in instructions of type I and type R?
   (c) Suppose the PC is set to 0x30000000. What range of address can be reached using the MIPS *branch if equal* (beq) instructions? Explain with reasoning.
   (d) Find out the width of the control memory of a horizontal microprogramming control unit for the following configuration:
   11 control lines for the processor of ALU and 32 registers, for conditional branching facility 8-bit status and provision to hold 128 words in the control memory.
   (e) Assume that variables *m, n, p, r* and *s* are assigned to register $s0, $s1, $s2 and $s3 and $s4 of a C program. In addition, base address of arrays *C* and *D* are in register $s7 and $s6. Translate the following MIPS code to C.

```
addi $t0, $s6, 4
add $t1, $s6, $zero
sw $t1, 0($t0)
lw $t0, 0($t0)
add $s0, $t1, $t0
add $s0, $s0, $t0
```

[**Marks: 5 + 2 + 5 + 5 + 5 = 22**]

3. Design [draw the datapath] the 16-bit Multi-Cycled single-purpose processor which executes/calculates the polynomial: $Y_i = a * X_i * X_i * X_i + b * X_i * X_i + c * X_i + d$, where $0 \leq i < 10$. Consider that a, b, c and d are constants and hold the values 2. X's 10 different values are stored in the 10 locations of memory (byte addressable) and also Y's values will be stored in memory. X_i and Y_i take 16-bits to store them into the memory. The "**Stop**" flag will be activated when ten Ys are calculated. Highlight & count the control signals in the datapath and design the controller [**table only**] for such processor. In the beginning of the execution, the processor will load the values in the corresponding datapath storage units.

[**Marks: 25**]

4. *lw* is the instruction with the longest latency on the Single-Cycled MIPS processor. We want to improve the clock-cycle time of the processor further by calculating the effective address early using other instructions. Meaning there is no offset in the **modified** lw and sw instructions. Let's consider the names of such instructions are *ld <dstn_reg>, <address_in_reg>* and *sd <src_reg>, <address_in_reg>*. Though it will increase the number of instructions of a program.

Parameter	Delay (ps)
t_{pcq_PC}	30
t_{mem}	250
t_{RFread}	150
t_{ALU}	200
t_{mux}	25
$t_{RFwrite}$	20

Critical path of lw

$$(T_c) = t_{pcq\_PC} + t_{mem} + \max\{t_{RFread},\ t_{select} + t_{mux}\} + t_{ALU} + t_{mem} + t_{mux} + t_{RFwrite}$$

Make similar assumption as discussed in the class.

(a) What would the revised clock cycle time be? Justify your answer.
(b) On this new Single-Cycled MIPS processor, would a program run faster or slower with the instruction mix is like: *50% R-type/I-type (non-lw), 26% lw, 12% sw and 12% beq*? By how much?
(c) Which factor determines whether a program will run faster or slower on the new Single-Cycled MIPS processor?
(d) Draw the modified datapath. More precisely, only the modified part **not the entire datapth**.

[**Marks: 3 + 5 + 3 + 3 = 14**]

5. We want to add the swap instruction, ***swap rs, rt***, in Multi-Cycled MIPS microprocessor. The interpretation of the instructions is `Reg[rs] = Reg[rt]; Reg[rt] = Reg[rs].`
   (a) Show the modification in the microprocessor's datapath. More precisely, only the modified part **not the entire datapth**.
   (b) Show the modification in the controller, i.e., only the table for the new instructions.

[**Marks: 3 + 3 = 6**]

6. The MIPS pipelined microprocessor is implemented in a basic form that does not handle data hazards (that is, the programmer must insert NOP instructions where necessary). To handle data hazards correctly, a typical *n*-instruction program needs 0.4**n* NOP instructions after optimization.
   (a) The cycle time of this pipeline without forwarding is 250 ps. Furthermore, suppose that adding forwarding hardware will reduce the number of NOPs from 0.4**n* to 0.05**n*, but increase the cycle time to 300 ps. What is the speedup of this new pipeline compared to the one without forwarding? **Show the detailed calculation.**
   (b) NOPs are required in different amounts for different programs. In order for the program to run faster on the pipeline with forwarding, how many NOPs (as a percentage of code instructions) must it have at minimum? **Show the detailed calculation.**
   (c) Consider the following sequence of code is running on 5-stages pipelined MIPS microprocessor. (1) The microprocessor has no forwarding unit, insert NOPs to ensure correct execution. (2) Now change and/or rearrange the code to minimize the number of NOPs needed.

[**Marks: 2 + 5 + 3 + 2 = 12**]

```
add $s3, $s1, $s0
lw   $s3, 4 ($s3)
lw   $s1, 0 ($s4)
or   $s2, $s3, $s2
sw  $s2, 0 ($s3)
```

-: End :-