Date: 20/03/25	Quiz 4	Duration : 40 min.
IIT Bombay	EE309 - Microprocessors	Total marks : 12

- Please maintain academic integrity.
- Show your work to get credits and state any assumptions you make.

---

1. [2 marks] A program runs in 10 s on a particular machine, with 70% of time spent doing multiply operations. We want to redesign the machine to provide it with faster multiply hardware. How much faster should the multiplier become, for the program to run 2 times faster ?

> **Solution:** f = 70; s = 2;
> $f_{frac} = f/100$; $p = (s * f_{frac})/(1 - s + s * f_{frac})$;
> Using f and s, we have $p = 3.5$.

2. [1 mark] Consider the three aspects: instruction count (IC), cycles-per instruction (CPI), and time per instruction (Cycle-time) that determine execution time. If we modify the processor to have a higher clock-frequency, which among the three aspects will be affected ?

> **Solution:** CPI and Cycle-time:w

3. [2 marks] Consider the given MiniMIPS code. Assume that `$a0` and `$a1` are used for the input and both initially contain the integers 3 and 8, respectively. Assume that `$v0` is used for the output. What will be the value stored in `$v0` ? Assume no overflows occur.

```
        add $t0, $zero, $zero
loop:   beq $a1, $zero, finish
        add $t0, $t0, $a0
        sub $a1, $a1, 1
        j loop
finish: addi $t0, $t0, 8
        add $v0, $t0, $zero
```

> **Solution:** Value stored in $v0$ is $3 * 8 + 8 = 32$

4. [2 marks] Complete the MiniMIPS code given below, which divides the value stored in `$s1` by that in `$s2` and stores the quotient in `$s0`. Note: Only fill in the four blanks.

```
       addi $s0, $zero, 0
loop:  slt $t0, $s2, $s1
       ____ $t0, $zero, finish
       ____ $s0, $s0, 1
       sub $s1, ____ , _____
       j loop
finish:
```

Solution:

```
       addi $s0, $zero, 0
loop:  slt $t0, $s2, $s1
       bne $t0, $zero, finish
       addi $s0, $s0, 1
       sub $s1, $s1 , $s2
       j loop
finish:
```

5. [5 marks] The single-cycle datapath and next address block that we discussed in class are shown in Figures 1 and 2. Now, consider the instruction: `lui rt,imm`.

```
Syntax: lui rt, imm
Action:
        rt <- (imm, 0x0000)
```

   (a) [1 mark] Write down the binary representation (in Hex) for the instruction, if register indices of `rs`, `rt` are 5, 6, respectively and `imm = 0x2020`.

   Solution:

   - You can use the reference sheet for R-format instruction [op, rs, rt, rd, sh, fn]
   - For `lui` instruction, $op = 001111$, $imm = 0010000000100000$
   - For registers, $rs = 00101$, $rt = 00110$
   - Together we have [001111 00101 00110 0010 0000 0010 0000] in binary
   - To have in Hex [0011 1100 1010 0110 0011 1000 0010 0100]
   - Binary representation in Hex 0x3CA62020

   (b) [4 marks] Derive/justify and write down the various control signal values for executing the `lui` instruction and show your results in Table 1 with the appropriate control values. Mention/show modifications (if any) needed to the datapath. The table values can be 0, 1, 2, 3 or X (don't care). A 0 or 1 that could have been an X counts as incorrect. In case the control signal is not for a `Mux`, then 1 implies setting it and 0 implies resetting, eg., `DataWrite = 1` implies data will be written into data memory. Just describe the specific ALU function needed against `ALUFunc`.

Table 1: Control values to execute `lui`

BRType	PCSrc	RegDst	RegWrite	ALUSrc	DataWrite	DataRead	RegInSrc
0	0	0	1	1*	0	0	1

ALUSrc: We have to send the imm field without SE to the ALU. The ALU will not use the rs input.
ALUFunc: Need to implement a left shift by 16-bits of the input imm field.

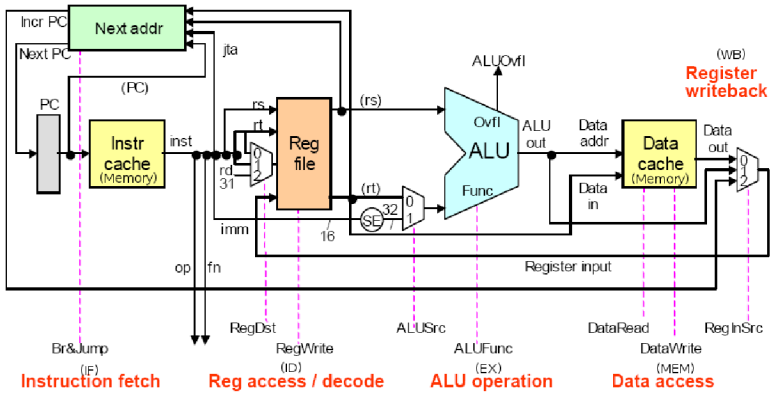

Figure 1: Single-cycle datapath. (Adapted from *Computer Architecture* by Behrooz Parhami.)

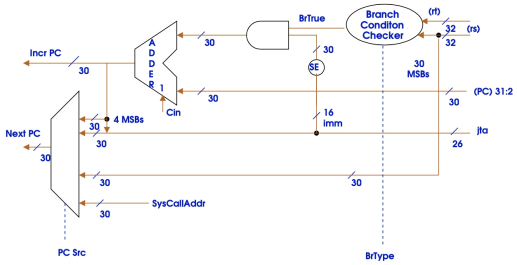

Figure 2: Next address block in single-cycle datapath.

Solution: We go from left to right in the datapath stages of instruction fetch (IF), instruction decode (ID), register access (ID), ALU operation (EX), Data access (MEM), and Register write back (WB) and note down the requirements and control signals.

- In IF, since this is a I-type instruction with no need for any specific next address decision, and we only need PC + 1. So we have, `BrType = 0`, `PCSrc = 0`.
- After the instruction decode, we know that source registers *rs* and *rt* need not be read. The destination register to be written is *rt* and hence we have `RegDst = 0` and `RegWrite = 1`. Note that the register write will happen only after the ALU execution, but the control block will have this enabled.
- For the EX stage, one input to the ALU is the *imm* field. We need left shift this value by 16-bits.
- For the **lui** instruction, the 'Data Cache' need not be read or written. Hence, `DataRead = 0` and `DataWrite = 0`.
- For the WB stage, the output from the 'ALU out' is to be written back. Hence, we need `RegInSrc = 1`.
- For the WB stage, the flags already set.