**CSE 3038 – Computer Organization** **Spring 2020**
**Prof. Haluk Topcuoglu**

# HW2 – Due 30 / 05 / 2020 23:00pm (No late submission)

# Questions

## Q.1 - 4.6 from 5th edition (4.6.1, 4.6.2, 4.6.3 included)

**4.6** When silicon chips are fabricated, defects in materials (e.g., silicon) and manufacturing errors can result in defective circuits. A very common defect is for one wire to affect the signal in another. This is called a cross-talk fault. A special class of cross-talk faults is when a signal is connected to a wire that has a constant logical value (e.g., a power supply wire). In this case we have a stuck-at-0 or a stuckat-1 fault, and the affected signal always has a logical value of 0 or 1, respectively. The following problems refer to bit 0 of the Write Register input on the register file in Figure 4.24.

**4.6.1** Let us assume that processor testing is done by filling the PC, registers, and data and instruction memories with some values (you can choose which values), letting a single instruction execute, then reading the PC, memories, and registers. These values are then examined to determine if a particular fault is present. Can you design a test (values for PC, memories, and registers) that would determine if there is a stuck-at-0 fault on this signal?

**Answer:**

To test for a stuck-at-0 fault on a wire, we need an instruction that puts that wire to a value of 1 and has a different result if the value on the wire is stuck at zero:

If this signal is stuck at zero, an instruction that writes to an odd-numbered register will end up writing to the even-numbered register. So if we place a value of zero in R30 and a value of 1 in R31, and then execute ADD R31, R30, R30 the value of R31 is supposed to be zero. If bit 0 of the Write Register input to the Registers unit is stuck at zero, the value is written to R30 instead and R31 will be 1.

**4.6.2** Repeat 4.6.1 for a stuck-at-1 fault. Can you use a single test for both stuck-at-0 and stuck-at-1? If yes, explain how; if no, explain why not.

**Answer:**

The test for stuck-at-zero requires an instruction that sets the signal to 1, and the test for stuck-at-1 requires an instruction that sets the signal to 0. Because the signal cannot be both 0 and 1 in the same cycle, we cannot test the same signal simultaneously for stuck-at-0 and stuck-at-1 using only one instruction.

The test for stuck-at-1 is analogous to the stuck-at-0 test:
We can place a value of zero in R31 and a value of 1 in R30, then use ADD R30,R31,R31 which is supposed to place 0 in R30. If this signal is stuck-at-1, the write goes to R31 instead, so the value in R30 remains 1.

**4.6.3** If we know that the processor has a stuck-at-1 fault on this signal, is the processor still usable? To be usable, we must be able to convert any program that executes on a normal MIPS processor into a program that works on this processor. You can assume that there is enough free instruction

memory and data memory to let you make the program longer and store additional data. Hint: the processor is usable if every instruction "broken" by this fault can be replaced with a sequence of "working" instructions that achieve the same effect.

## Answer:

We need to rewrite the program to use only odd-numbered registers

## Q.2 - 4.8 from 5th edition (4.8.1, 4.8.2, 4.8.3, 4.8.4, 4.8.6 included)

**4.8** In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

| IF | ID | EX | MEM | WB |
|---|---|---|---|---|
| 250ps | 350ps | 150ps | 300ps | 200ps |

Also, assume that instructions executed by the processor are broken down as follows:

| alu | beq | lw | sw |
|---|---|---|---|
| 45% | 20% | 20% | 15% |

**4.8.1** What is the clock cycle time in a pipelined and non-pipelined processor?

| Pipelined | Single-cycle |
|---|---|
| 350 ps | 1250 ps |

**4.8.2** What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

| Pipelined | Single-cycle |
|---|---|
| 1750 ps | 1250 ps |

**4.8.3** If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

| Stage to split | New clock cycle time |
|---|---|
| ID | 300 ps |

**4.8.4** Assuming there are no stalls or hazards, what is the utilization of the data memory?

| | |
|---|---|
| a. | 35% |

**4.8.6** Instead of a single-cycle organization, we can use a multi-cycle organization where each instruction takes multiple cycles but one instruction finishes before another is fetched. In this organization, an instruction only goes through stages it actually needs (e.g., ST only takes 4 cycles because it does not need the WB stage). Compare clock cycle times and execution times with single-cycle, multi-cycle, and pipelined organization.

**Answer:**

We already computed clock cycle times for pipelined and single cycle organizations, and the multi-cycle organization has the same clock cycle time as the pipelined organization. We will compute execution times relative to the pipelined organization. In single-cycle, every instruction takes one (long) clock cycle. In pipelined, a long-running program with no pipeline stalls completes one instruction in every cycle. Finally, a multicycle organization completes a LW in 5 cycles, a SW in 4 cycles (no WB), an ALU instruction in 4 cycles (no MEM), and a BEQ in 3 cycles (no MEM, no WB). So we have the speedup of pipeline

| Multi-cycle execution time is X times pipelined execution time, where X is: | Single-cycle execution time is X times pipelined execution time, where X is: |
|---|---|
| 0.20*5 + 0.60*4 + 0.20*3 = 4 | 1250 ps/350 ps = 3.57 |

**Q.3** -We wish to add the instruction **addi (add immediate)** to the single-cycle datapath described in the chapter. Add any necessary datapaths and control signals to the single-cycle datapath of the Figure 4.24 which represents the datapath and show the necessary additions (if any) to the Figure 4.18 which represents the setting of the control lines.

**Answer**:

No datapath changes are required.

**Q.4.** - Assume that you are required to extend our single-cycle MIPS implementation so that it handles the "jnew" instruction, where "jnew" has R-type instruction format where rd field is 0. An example is given below:

Example: jnew $s3, $s4 # unconditionally jump to the address given in Memory[$s3+$s4]

# PC← Memory[$s3+$s4]

**Explain your design** by listing the required changes in the complete single-cycle datapath clearly given in Figure 4.24 (additional wires, muxes and control and selector signals if necessary).

**Answer**:

1. Connect "Read Data" output of data memory to the multiplexor controlled with "Jump" control signal.
2. Now the multiplexor (control with jump) will have 2 control inputs
   (Jump0)->original one (Jump1)->new one

**For the homework, you have to work alone and submit your own work. In case of any form of copying all parties will get 0 grade.**