# CENG3420 Homework 2

**Due**: Mar. 06, 2018

Please submit PDF or WORD document directly onto blackboard.
DO NOT SUBMIT COMPRESSED ZIP or TARBALL.

## Solutions

**Q1** (15%) The basic single-cycle MIPS implementation in Figure 1 can only implement some instructions. New instructions can be added to an existing Instruction Set Architecture. Following questions refer to the new instruction:

Instruction `LWI Rt, Rd(Rs)`

Interpretation `Reg[Rt] = Mem[Rd + Reg[Rs]]`

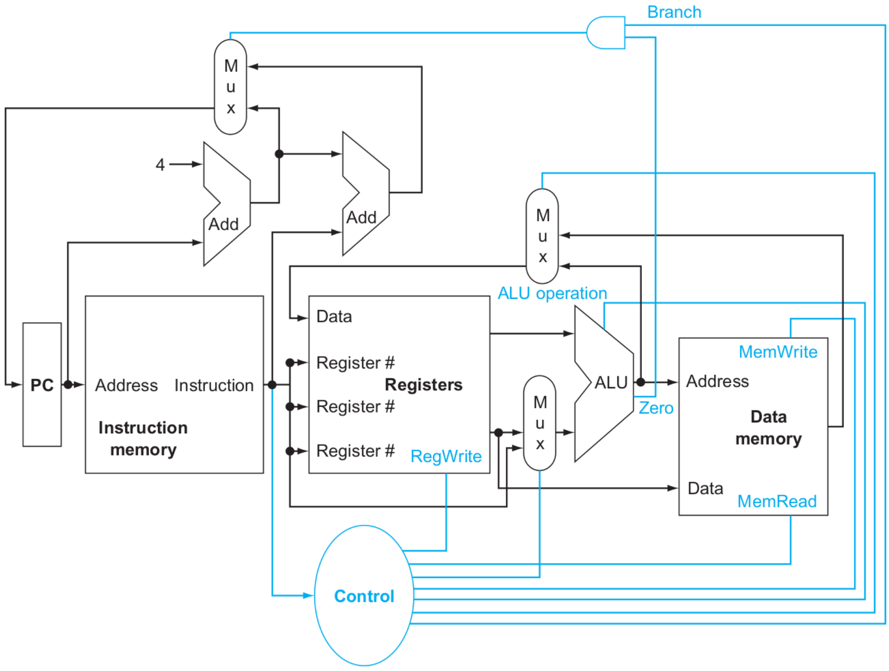

Figure 1: The basic implementation of the MIPS subset, including the necessary multiplexors and control lines.

1. Which existing blocks (if any) can be used for this instruction?
2. Which new functional blocks (if any) do we need for this instruction?
3. What new signals do we need (if any) from the control unit to support this instruction?

**A1**

1. Instruction memory, one register read ports, the path that passed the immediate to the ALU, and the register write port.
2. We need to extend the existing ALU to also do shifts (SLL, to extend the offset to 32bit value).

3. We need to change the ALU operation control signals to support the SLL operation in the ALU.

**Q2** (15%) Following problems assume that logic blocks needed to implement a processor's datapath have the following latencies (Table 1):

Table 1: Question 2

| Item | I-Mem | Add | Mux | ALU | Regs | D-Mem | Sign-Extend | Shift-Left-2 |
|---|---|---|---|---|---|---|---|---|
| Latency (ps) | 200 | 70 | 20 | 90 | 90 | 250 | 15 | 10 |

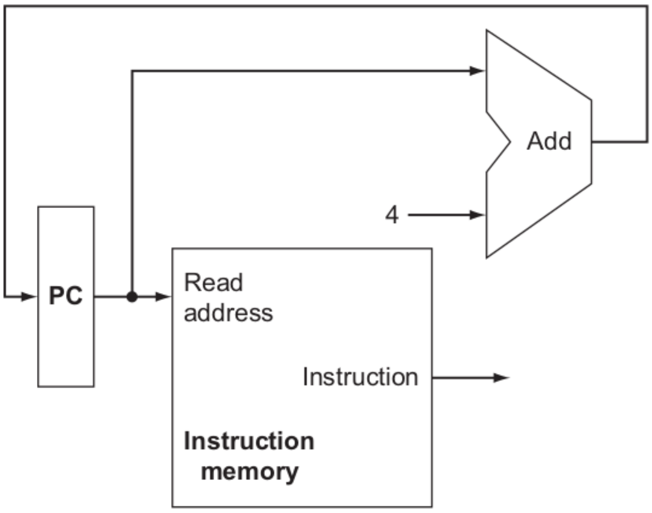

Figure 2: A portion of the datapath used for fetching instructions and incrementing the program counter.

1. If the only thing we need to do in a processor is fetch consecutive instructions (Figure 2), what would the cycle time be?
2. Consider a datapath similar to the one in Figure 3, but for a processor that only has one type of instruction: unconditional PC-relative branch. What would the cycle time be for this datapath?
3. Repeat 2, but this time we need to support only conditional PC-relative branches.

**A2**

1. $200ps$.
2. Critical path include: Instruction memory, Sign-extend, Shift left 2, Add and Mux. The cycle time will be

$$CycleTime = 200 + 15 + 10 + 70 + 20 = 315ps. \tag{1}$$

An example of this problem can be found at slides L06.12, L06.13. Because we are talking about unconditional branch, no need to access Register File and ALU. Note that the architecture in Fig 3 is slightly different from L06.13 with an additional MUX between Add and PC.

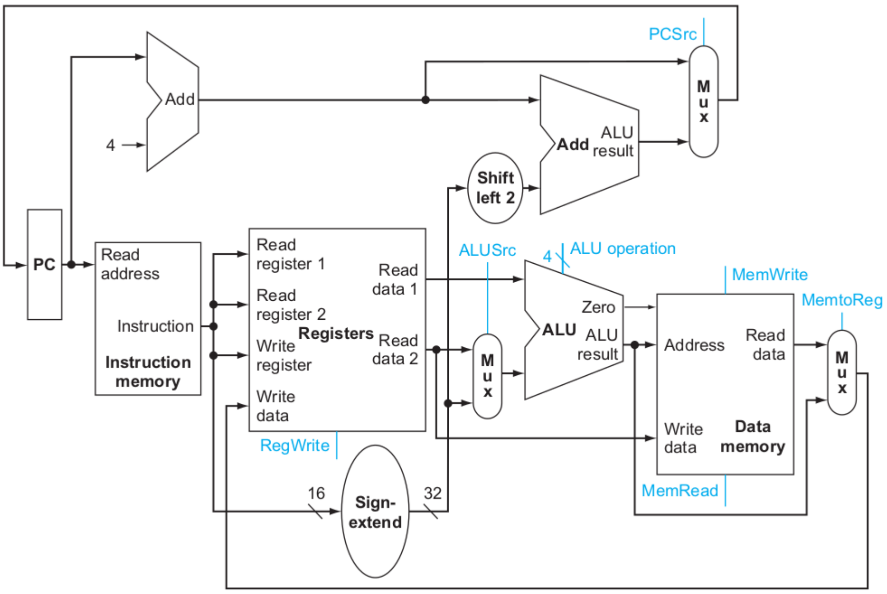

Figure 3: The simple datapath for the core MIPS architecture combines the elements required by different instruction classes.

3. For the PC-relative conditional branch, there are two sub-datapath to finish the instruction before entering the final MUX. (1) IM→Sign-ext→ Shift left 2→ ADD and (2) IM→Register File→MUX→ALU. Then,

$$Path_1 = 200 + 15 + 10 + 70 = 295ps \quad (2)$$
$$Path_2 = 200 + 90 + 20 + 90 = 400ps > Path_1. \quad (3)$$

Thus, the cycle time is determined by the longest path,

$$CycleTime = Path_2 + MUX = 420ps. \quad (4)$$

**Q3** (15%) Given the following specs of the datapath latencies:

| Stages | IF | ID | EX | MEM | WB |
|---|---|---|---|---|---|
| Latencies (ps) | 200 | 170 | 220 | 210 | 150 |

1. What is the clock cycle time in a pipelined and non-pipelined processor?
2. What is the total latency of an `LW` instruction in a pipelined and non-pipelined processor?
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?

**A3**
1. Non-pipelined: $950ps$; Pipelined: $220ps$.
2. Non-pipelined: $950ps$; Pipelined: $1100ps$.

3. Split EX stage. New clock cycle will be 210*ps*.

**Q4** (15%) Regarding the following instructions:

```
I1:  lw R1, 40(R2)
I2:  add R2, R3, R3
I3:  sw R2, 50(R1)
```

1. Indicate dependencies and their type.
2. Assume there is no forwarding in this pipelined processor. Add `NOP` instructions to eliminate hazards.
3. Assume there is full forwarding. Indicate hazards and add `NOP` instructions to eliminate them.

**A4**

1. RAW: `R1` from `I1` to `I3`; `R2` from `I2` to `I3`. WAR: `R2` from `I1` to `I2` and `I3`.
2. Add one `nope` after `I2`.
3. No hazard because of the existence of forwarding.s

**Q5** (10%) Assume it takes one clock to send address to DRAM memory and one clock to send data back. DRAM has 8 cycle latency for first byte, and 4 cycles for each of subsequent bytes in the block. To transfer a 8-byte block, calculate the cycle number if we need:

1. non-interleaving;
2. 2-module interleaving.

**A5**

1. 38
2. As shown in the following figure, the total cycle number is 23.

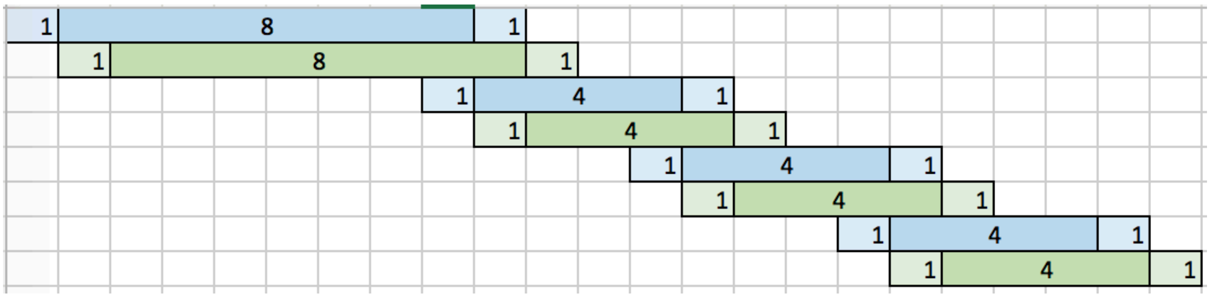

Figure 4: A5

**Q6** (15%) In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where elements within the same row are stored contiguously.

```
for (J=0; J<8000; J++)
        for (I=0; I<8; I++)
                A[I][J]=B[J][I]+A[J][I];
```

1. How many 32-bit integers can be stored in a 16-byte cache line?
2. References to which variables exhibit temporal locality?
3. References to which variables exhibit spatial locality?

**A6**

1. 4
2. I, J
3. A[J][I], B[J][I]

**Q7** (10%) For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache (as in Table 2).

Table 2: Q7

| | Tag | Index | Offset |
|---|---|---|---|
| a | 31-10 | 9-5 | 4-0 |
| b | 31-12 | 11-6 | 5-0 |

1. What is the cache line size (in word)?
2. What is the ratio between total bits required for such a cache implementation over the data storage bits?

**A7**

1. a. $2^5/4 = 8$; b. $2^6/4 = 16$.
2. a. line size is 256 bits. Total bit count is 256+22+1=277. Ratio is 279/256=1.09. b. Ratio is 1.04.

**Q8** (5%) Describe two cache replacement strategies.

**A8**

1. Write-Through : always write the data into both the cache block and the next level in the memory hierarchy.
2. Write-Back: Write to memory hierarchy when that cache block is “evicted”.