CENG3420 Homework 2

Due: Mar. 02, 2019

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]]`

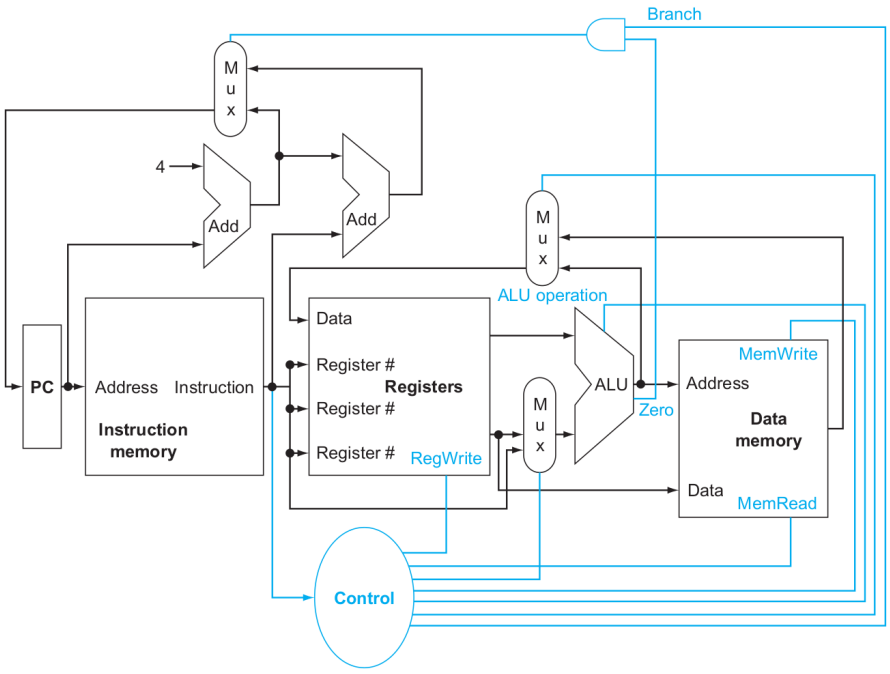

Figure 1: The basic implementation of the MIPS subset, including the necessary multiplexors and control lines.

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)	750	200	50	250	300	500	100	0

1. If the only thing we need to do in a processor is fetch consecutive instructions (Figure 2), what would the cycle time be?

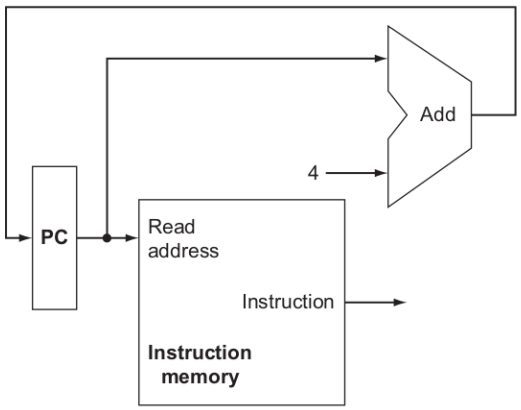

Figure 2: A portion of the datapath used for fetching instructions and incrementing the program counter.

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?

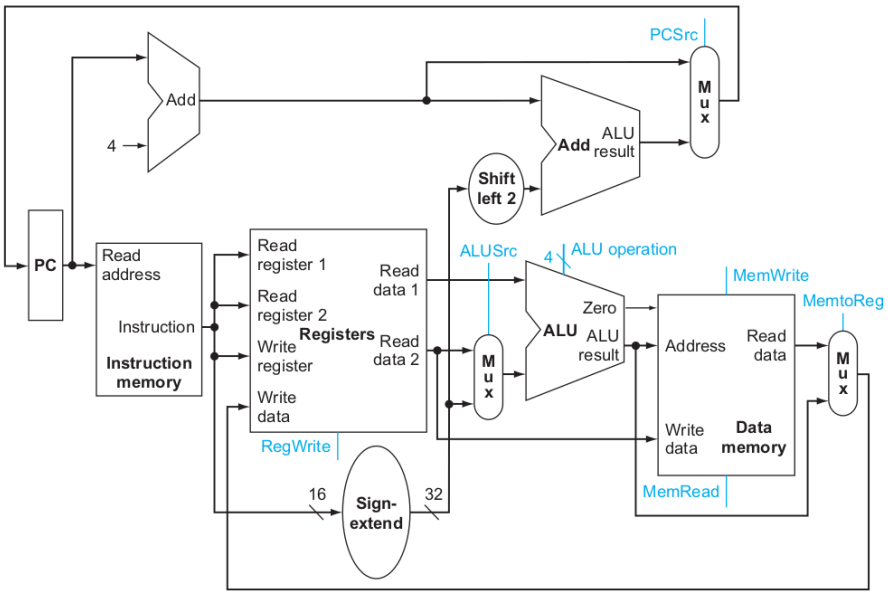

Figure 3: The simple datapath for the core MIPS architecture combines the elements required by different instruction classes.

3. Repeat 2, but this time we need to support only conditional PC-relative branches.

A2 1. $750ps$.

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

$$CycleTime = 750 + 100 + 0 + 200 + 50 = 1100ps. \quad (1)$$

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,

$$Path_1 = 750 + 100 + 0 + 200 = 1050ps \quad (2)$$
$$Path_2 = 750 + 300 + 50 + 250 = 1350ps > Path_1. \quad (3)$$

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

$$CycleTime = Path_2 + MUX = 1400ps. \quad (4)$$

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 210ps.

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 `nop` after `I2`.

3. No hazard because of the existence of forwarding.

Q5 (10%) Regarding the following MIPS instruction:

```
sw R16, -100(R6)
```

1. Which registers need to be read, and which registers are actually read?
2. What does this instruction do in the EX and MEM stages?

A5 1. `R6` and `R16` need to be read; `R6` and `R16` are actually read.

2. EX stage: -100 + `R6`; MEM stage: write value to memory.

Q6 (15%) Given the following loop, assume that perfect branch prediction is used (no stalls due to control hazards), that there are no delay slots, and that the pipeline has full forwarding support. Also assume that many iterations of this loop are executed before the loop exits.

```
loop:  add R1, R2, R1
       lw  R2, 0(R1)
       lw  R2, 16(R2)
       slt R1, R2, R4
       beq R1, R9, loop
```

1. Show a pipeline execution diagram for the third iteration of this loop, from the cycle in which we fetch the first instruction of that iteration up to (but not including) the cycle in which we can fetch the first instruction of the next iteration. Show all instructions that are in the pipeline during these cycles (not just those from the third iteration).

Loop	
2: LW R2,16(R2) 2: SLT R1,R2,R4 2: BEQ R1,R9,Loop 3: ADD R1,R2,R1 3: LW R2,0(R1) 3: LW R2,16(R2) 3: SLT R1,R2,R4 3: BEQ R1,R9,Loop	WB EX MEM WB ID EX MEM WB IF ID EX MEM WB IF ID EX MEM WB IF ID *** EX MEM IF *** ID *** IF ***

Figure 4: Answer of Q6-1

2. How often (as a percentage of all cycles) do we have a cycle in which all five pipeline stages are doing useful work?
3. At the start of the cycle in which we fetch the first instruction of the third iteration of this loop, what is stored in the IF/ID register?

A6 1. See the Figure 4 below:

2. Since there is 7 cycles per Loop Iteration, and 1 cycle in which all stages do useful work, the ration is $1/7 = 14\%$.
3. The address of that first instruction of the third iteration ($PC + 4$ for the `BEQ` from the previous iteration) and the instruction word of the `BEQ` from the previous iteration.

Q7 (15%) Regarding the following instruction sequences:

```
add R1, R2, R1
lw  R2, 0(R1)
lw  R1, 4(R1)
or  R3, R1, R2
```

1. Find all data dependences in this instruction sequence.
2. Find all hazards in this instruction sequence for a 5-stage pipeline with and then without forwarding.
3. To reduce clock cycle time, we are considering a split of the MEM stage into two stages. Repeat 2 for this 6-stage pipeline.

A7 1. See Figure 5:

Instruction Sequence	RAW	WAR	WAW
I1: ADD R1,R2,R1 I2: LW R2,0(R1) I3: LW R1,4(R1) I4: OR R3,R1,R2	(R1) I1 to I2, I3 (R2) I2 to I4 (R1) I3 to I4	(R2) I1 to I2 (R1) I1, I2 to I3	(R1) I1 to I3

Figure 5: Answer of Q7-1

2. Only RAW dependences can become data hazards. With forwarding, only RAW dependences from a load to the very next instruction become hazards. Without

forwarding, any RAW dependence from an instruction to one of the following 3 instructions becomes a hazard (see Figure 6).

Instruction Sequence	With Forwarding	Without Forwarding
I1: ADD R1,R2,R1 I2: LW R2,0(R1) I3: LW R1,4(R1) I4: OR R3,R1,R2	(R1) I3 to I4	(R1) I1 to I2, I3 (R2) I2 to I4 (R1) I3 to I4

Figure 6: Answer of Q7-2

3. With forwarding, only RAW dependences from a load to the next two instructions become hazards because the load produces its data at the end of the second MEM stage. Without forwarding, any RAW dependence from an instruction to one of the following 4 instructions becomes a hazard (see Figure 7).

Instruction Sequence	With Forwarding	RAW
I1: ADD R1,R2,R1 I2: LW R2,0(R1) I3: LW R1,4(R1) I4: OR R3,R1,R2	(R2) I2 to I4 (R1) I3 to I4	(R1) I1 to I2, I3 (R2) I2 to I4 (R1) I3 to I4

Figure 7: Answer of Q7-3