CENG3420 Homework 2

Due: Apr. 07, 2020

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 `sw Rs, Im(Rt)`

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

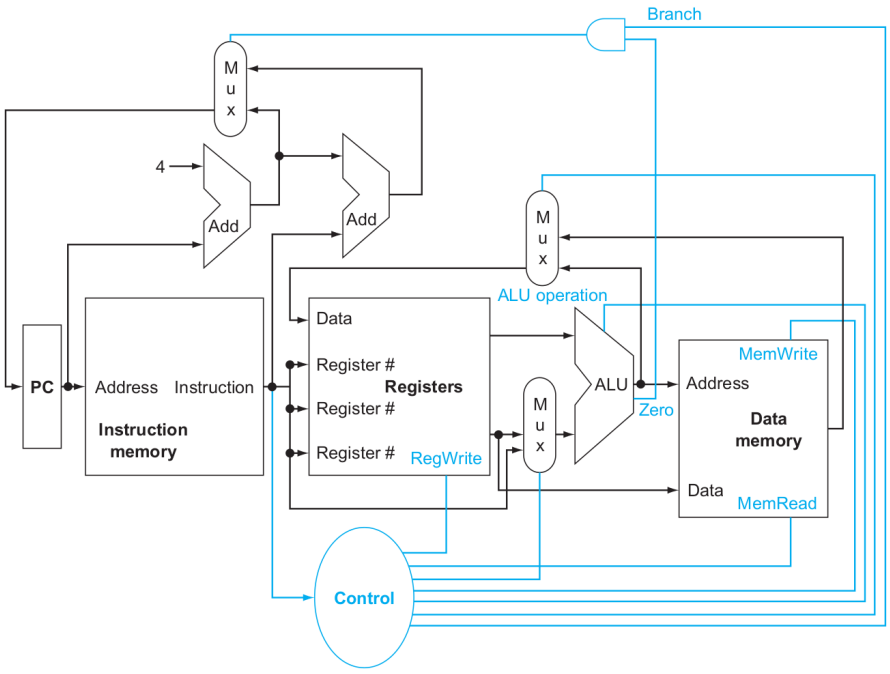

Figure 1: The basic implementation of the MIPS subset, including the necessary multiplexors and control lines.

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 with sign extended block to also do shifts (SLL and SRA, 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)	350	100	20	120	120	400	20	5

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

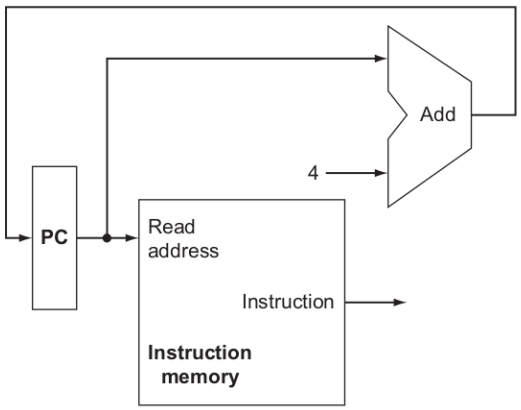

Figure 2: A portion of the datapath used for fetching instructions and incrementing the program counter.

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? Destribe the critical path and the function of each unit.

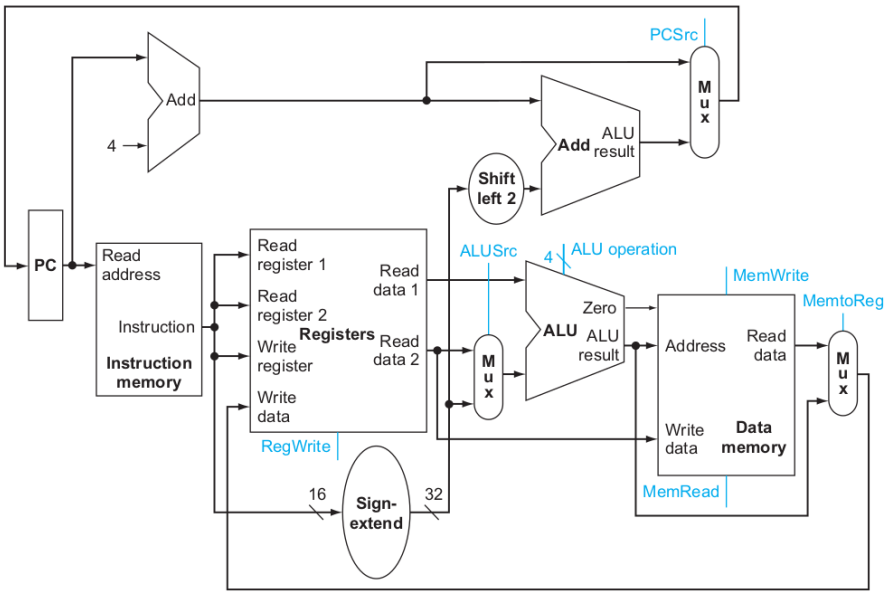

Figure 3: The simple datapath for the core MIPS architecture combines the elements required by different instruction classes.

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

A2 1. I-Mem takes longer than the Add unit, so the clock cycle time is equal to the latency of the I-Mem: $350ps$.

2. Because the latency of I-Mem is longer than the latency of the Add unit. The path for computing PC+4 is shorter than jumping to the target location. The critical path for this instruction is through the Instruction memory, Sign-extend, Shift left 2 to get the offset, the top Add unit to compute the new PC, and Mux to select that value instead of PC+4. The cycle time will be

$$CycleTime = 350 + 20 + 5 + 100 + 20 = 495ps. \quad (1)$$

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,

$$Path_1 = 350 + 20 + 5 + 100 = 475ps \quad (2)$$
$$Path_2 = 350 + 120 + 20 + 120 = 610ps > Path_1. \quad (3)$$

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

$$CycleTime = Path_2 + Mux = 630ps. \quad (4)$$

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

Stages	IF	ID	EX	MEM	WB
Latencies (ps)	200	160	250	200	140

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: $250ps$.

2. Non-pipelined: $950ps$; Pipelined: $1250ps$.

3. Split EX stage. New clock cycle will be $200ps$.

Q4 (10%)

alu	beq	lw	sw
50%	25%	15%	10%

1. Assuming there are no stalls or hazards, what is the utilization of the data memory?
2. Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

A4 1. Data memory is utilized by lw and sw instructions in the MIPS ISA. So the utilization is 25% of the clock cycles.

2. The write port is utilized by alu and lw instructions. The utilization is 65% of the clock cycles.

Q5 (15%) You are required to develop some simple measures of pipeline performance and relative speedup.

1. Let $T_{k,n}$ be the total time required for a pipeline with k stages to execute n instructions. Speedup of k stage pipeline is given by,

$$S_k = \frac{T_{1,n}}{T_{k,n}}. \tag{5}$$

Determine S_k in terms of k and n.

2. Consider an instruction sequence of length n that is streaming through the instruction pipeline. Let p be the probability of encountering a conditional or unconditional branch instruction, and let q be the probability that execution of a branch instruction I causes a jump to a nonconsecutive address. Assume that each such jump requires the pipeline to be cleared, destroying all ongoing instruction processing, when I emerges from the last stage. Determine S_k in terms of k, n, p and q.

A5 1.

$$S_k = \frac{nk}{k+n-1}.$$

2.

$$S_k = \frac{nk}{pqnk + (1 - pq)(k + n - 1)}.$$

Q6 (15%) Given the following loop,

```
loop:  add R1, R2, R3
       sub  R4, R1, R3
       lw   R6, 0(R4)
       slt  R6, R7, R6
       beq  R1, R6, loop
```

1. 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. Show a pipeline execution diagram for the second 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.
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. ~~Show a pipeline execution diagram by inserting stalls to resolve data hazards.~~

Table 2: Answer of Q6-1

slt R6, R7, R6	MEM	WB					
beq R1, R6, loop	ID	EX	MEM	WB			
add R1, R2, R3	IF	ID	EX	MEM	WB		
sub R4, R1, R3		IF	ID	EX	MEM	WB	
lw R6, 0(R4)			IF	ID	EX	MEM	WB
slt R6, R7, R6				IF	ID	***	EX
beq R1, R6, loop						***	IF

A6

1. See Table 2. Stages in which the particular instruction is not doing useful work are marked in blue. Note that a BEQ instruction is doing useful work in the MEM stage, because it is determining the correct value of the next instruction's PC in that stage.
2. Since there is 7 cycles per Loop Iteration, and 0 cycle in which all stages do useful work, the ration is $0/7 = 0\%$.

Q7 (15%)

Table 3: Instruction Category

R-type	beq	jmp	lw	sw
40%	25%	5%	25%	5%

The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as Table 3. Also, assume that the branch predictor accuracies are as Table 4.

Table 4: Predictor Accuracy

Always-Taken	Always-Not-Taken	2-Bit
45%	55%	85%

1. Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the EX stage, that there are no data hazards, and that no delay slots are used.
2. Repeat 1 for the "always-not-taken" predictor.
3. Repeat 1 for for the 2-bit predictor.

A7

1. Each branch that is not correctly predicted by the always-taken predictor will cause 3 stall cycles, so we have: $3 * (1 - 0.45) * 0.25 = 0.41$.
2. Each branch that is not correctly predicted by the always-not-taken predictor will cause 3 stall cycles, so we have: $3 * (1 - 0.55) * 0.25 = 0.34$.
3. Each branch that is not correctly predicted by the 2-bit predictor will cause 3 stall cycles, so we have: $3 * (1 - 0.85) * 0.25 = 0.113$.