CS2100 Computer Organisation

Tutorial #5: MIPS Processor: Datapath and Control

Answers

Tutorial Questions

Questions 1 and 2 refer to the complete datapath and control design covered in lectures #11 and #12. Please use the diagram in Lecture #12 slide 29 or in the COD MIPS 4th edition textbook, Figure 4.17. For your convenience, the diagram is also included at the end of this tutorial sheet.

1. Let us perform a complete trace to understand the working of the complete datapath and control implementation. Given the following three hexadecimal representations of MIPS instructions:

   i. `0x8df80000: lw $24, 0($15)`
   ii. `0x1023000C: beq $1, $3, 12`
   iii. `0x0285c822: sub $25, $20, $5`

   For each instruction encoding, do the following:

   (a) Fill in the two tables below. The first table contains the various data (information) at the datapath elements, while the second table records the control signals generated. Use the notation $8 to represent register number 8, [$8] to represent the content of register number 8, and Mem(X) to represent the data at memory address X.

Registers File				**ALU**		**Data Memory**	
RR1	RR2	WR	WD	Opr1	Opr2	Address	Write Data

   [Wr = Write; Rd = Read; M = Mem; R = Reg]

RegDst	**RegWr**	**ALUSrc**	**MRd**	**MWr**	**MToR**	**Brch**	**ALUop**	**ALUctrl**

   (b) Indicate the value of the PC after the instruction is executed.

Answers:

Only values in RED and **BOLD** font are actually utilized in the execution.

i. `0x8df80000 = lw $24, 0($15);` next PC = PC+4

Registers File				ALU		Data Memory	
RR1	RR2	WR	WD	Opr1	Opr2	Address	Write Data
$15	$24	**$24**	**MEM([$15]+0)**	**[$15]**	**0**	**[$15]+0**	[$24]

RegDst	RegWr	ALUSrc	MRd	MWr	MToR	Brch	ALUop	ALUctrl
0	1	1	1	0	1	0	00	0010

ii. `0x1023000C = beq $1, $3, 12;` next PC = PC+4 *or* (PC+4)+(12×4)

Registers File				ALU		Data Memory	
RR1	RR2	WR	WD	Opr1	Opr2	Address	Write Data
$1	**$3**	$3 or $0	[$1]-[$3] *or* random value	**[$1]**	**[$3]**	[$1]-[$3]	[$3]

RegDst	RegWr	ALUSrc	MRd	MWr	MToR	Brch	ALUop	ALUctrl
X	0	0	0	0	X	1	01	0110

iii. `0x0285c822 = sub $25, $20, $5;` next PC = PC+4

Registers File				ALU		Data Memory	
RR1	RR2	WR	WD	Opr1	Opr2	Address	Write Data
$20	**$5**	**$25**	**[$20]-[$5]**	**[$20]**	**[$5]**	[$20]-[$5]	[$5]

RegDst	RegWr	ALUSrc	MRd	MWr	MToR	Brch	ALUop	ALUctrl
1	1	0	0	0	0	0	10	0110

2. With the complete datapath and control design, it is now possible to estimate the latency (time needed for a task) for the various type of instructions. Given below are the resource latencies of the various hardware components (ps = picoseconds = 10^{-12} second):

Inst-Mem	Adder	MUX	ALU	Reg-File	Data-Mem	Control/ ALUControl	Left-shift/ Sign-Extend/ AND
400ps	100ps	30ps	120ps	200ps	350ps	100ps	20ps

Give the estimated latencies for the following MIPS instructions:

(a) SUB instruction (e.g. **`sub $25, $20, $5`**)
(b) LW instruction (e.g. **`lw $24, 0($15)`**)
(c) BEQ instruction (e.g. **`beq $1, $3, 12`**)

What do you think the **cycle time** should be for this particular processor implementation?

Hint: First, you need to find out the **critical path** of an instruction, i.e. the path that takes the longest time to complete. Note that there could be several parallel paths working more or less simultaneously.

Answers:

(a) SUB instruction (R-type):

Critical Path:
I-Mem → Reg.File → MUX(ALUSrc) → ALU → MUX(MemToReg) → Reg.File

Note: I-MEM → Control is a parallel path, the earliest signal needed is the ALUSrc. So, as long as the Control latency is lesser than Reg.File access latency, then it will not be in the critical path. Once the signal is generated, the Control latency will no longer affect the overall delays.

Similarly, there is another path to calculate the next PC (I-MEM → Control → AND → MUX(PCSrc) which is again not critical to the overall latency.

Latency = 400 + 200 + 30 + 120 + 30 + 200 = 980ps

(b) LW instruction:

Critical Path:
I-Mem → Reg.File → ALU → DataMem → MUX(MemToReg) → Reg.File

Latency = 400 + 200 + 120 + 350 + 30 + 200 = 1300ps

Note: The path I-Mem → Immediate → MUX(ALUSrc) occurs simultaneously with the above.

(c) BEQ instruction:

Critical Path:
I-Mem → Reg.File → MUX(ALUSrc) → ALU → AND → MUX(PCSrc)

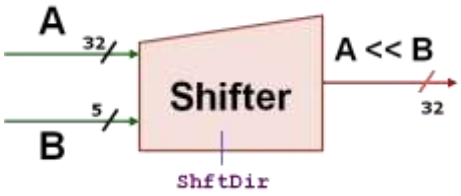
Latency = 400 + 200 + 30 + 120 + 20 + 30 = 800ps

Since LW has the longest latency. The overall cycle time of the whole machine is determined by LW, i.e. at least 1300ps.

3. [AY2011/12 Semester 2 Exam]
Let us consider the task of incorporating the "**`sll`**" (shift-left logical) instruction into the MIPS processor. Suppose we have the following new datapath element "Shifter" which has the following functionality:

Inputs:
A is a 32-bit value
B is a 5-bit unsigned value

Output:
A << B, i.e. A left shifted by B positions



Control (ShftDir):
0 for left-shift, 1 for right-shift
Hardwired to 0 for left-shifting for this part

(a) For the inputs A and B, indicate the correct values to connect to them.

For the output (A<<B), indicate **an existing multiplexer** that it can connect to. Briefly explain how to control that existing multiplexer properly. You do not need to figure out how the control signal(s) is/are generated.

(b) It is obviously a waste to use the "Shifter" element in (a) for left shifting only. Suppose we want to incorporate the "**`srl`**" (shift-right logical) instruction into the processor design in (a). Considering only the **ShftDir** control signal, explain how you can generate this signal from the Opcode/Funct Field of "**`sll`**" and "**`srl`**" instructions (given below). Give the simplest Boolean expression.

`sll`: opcode = 0_{16}; funct = 0_{16}

`srl`: opcode = 0_{16}; funct = 2_{16}

You may use Op[5:0] and F[5:0] to indicate the respective bits in the opcode and funct fields.

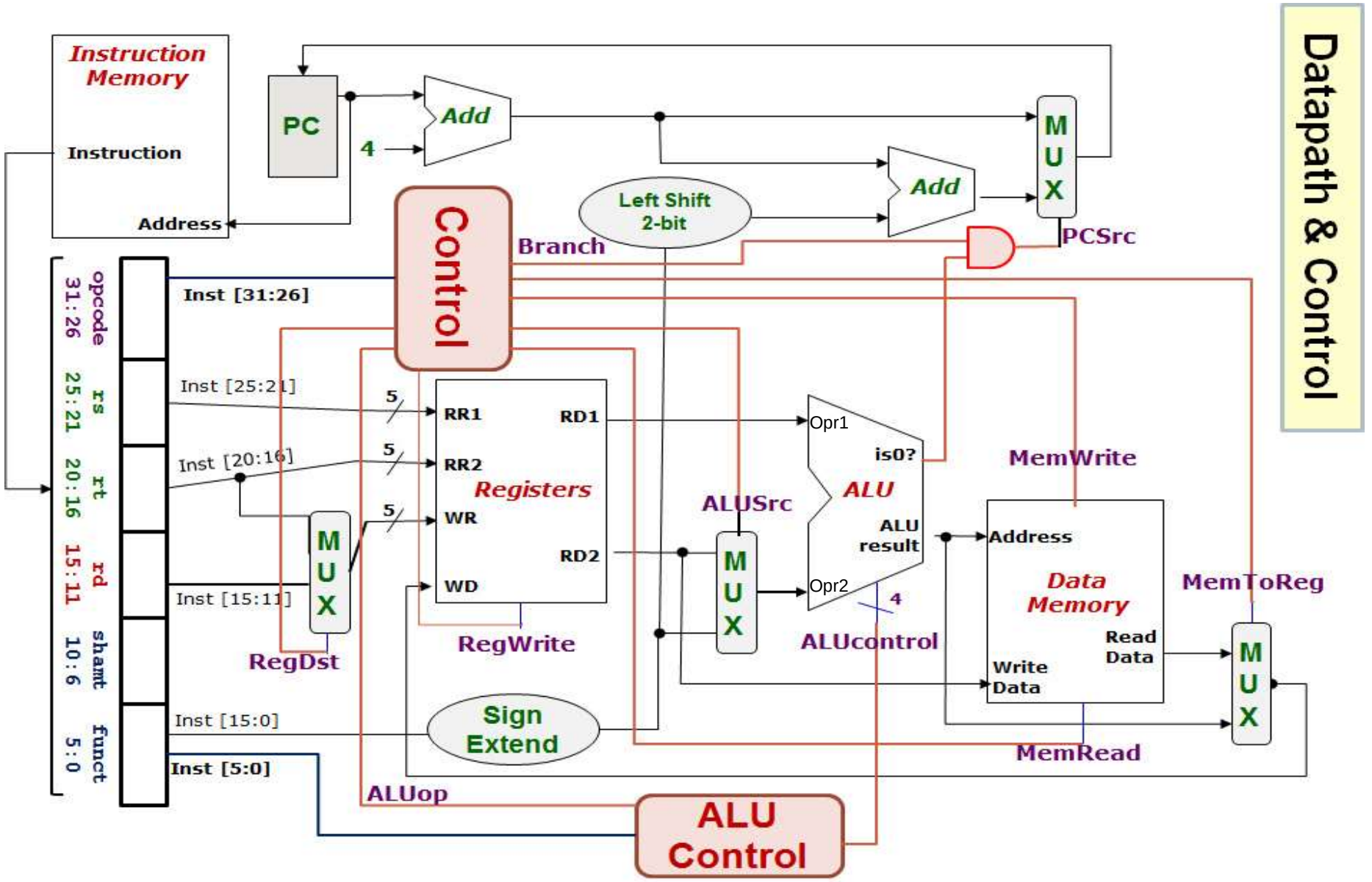
(c) How does the processor with the above new functionalities handle non-shift instructions? Briefly explain.

Answers:

(a) A = RD2; B = Inst[10:6].
Use the "MemToReg" multiplexer. Extend the multiplexer to a 4:1 MUX. The result to be written should be chosen from these 3 sources: Memory data, ALU results or Shifter Result.

(b) ShftDir = F[1] or Inst[1]

(c) One possible answer: As long as we control the MemToReg multiplexer correctly, non-shift instructions will not cause any problem. Non-shift instructions may generate results from the "Shifter" element, but the result won't be written as MemToReg determines the correct source to be used.


Datapath & Control
Instruction Memory
Instruction
Address
PC
4
Add
Add
M U X
Left Shift 2-bit
Branch
PCSrc
Control
opcode 31:26
rs 25:21
rt 20:16
rd 15:11
shamt 10:6
funct 5:0
Inst [31:26]
Inst [25:21]
Inst [20:16]
Inst [15:11]
Inst [15:0]
Inst [5:0]
5
RR1
RR2
WR
WD
RD1
RD2
Registers
RegWrite
RegDst
ALUSrc
Opr1
Opr2
is0?
ALU
ALU result
4
ALUcontrol
MemWrite
Address
Data Memory
Write Data
Read Data
MemRead
MemToReg
Sign Extend
ALUop
ALU Control