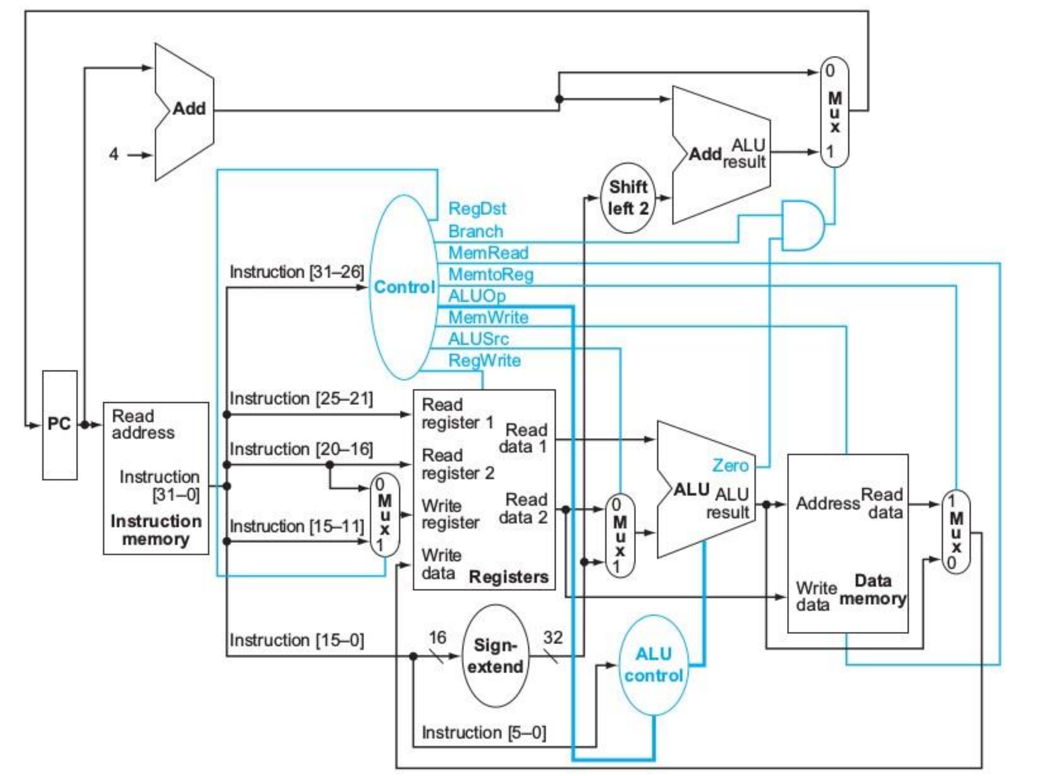
**CO200 – COA. Tutorial 6.** Duration: 60 minutes. Max: 29 – Sept – 2016




1. Consider the datapath shown. In this exercise we examine in detail how an instruction is executed in a single-cycle datapath. Problems in this exercise refer to a clock cycle in which the processor fetches the following instruction word: 10101100011000100000000000010100

Assume that data memory is all zeros and that the processor's registers have the following values at the beginning of the cycle in which the above instruction word is fetched:

| R0 | R1 | R2 | R3 | R4 | R5 | R6 | R8 | R12 | R31 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | -1 | 2 | -3 | -4 | 10 | 6 | 8 | 2 | -16 |

a. What are the outputs of the sign-extend and the jump "Shift left 2" unit for this instruction word?
sign-extend : 00000000000000000000000000010100
jump "Shift left 2": 0000000000000000000000001010000

b. What are the values of the ALU control unit's inputs for this instruction?
010100

c. What is the new PC address after this instruction is executed? Highlight the path through which this value is determined.
New PC: PC + 4
Path: PC to Add (PC + 4) to branch Mux to PC

d. For each Mux, write down the values of its data output during the execution of this instruction and the register values.
WrReg Mux: 2 or 0 (RegDst is X)
ALU Mux: 20
Mem/ALU Mux: X
Branch Mux: PC + 4

e. For the ALU and the two add units, what are their data input values?
ALU: -3 and 20
Add (PC+4): PC and 4
Add (Branch): PC+4 and 20×4

f. What are the values of all inputs for the "Registers" unit?

Read Register 1: 3
Read Register 2: 2
Write Register: X (2 or 0)
Write data: X
RegWrite: 0

2. As a designer you have to add a "Dependence Check Block (DCB)" into the 5 stage MIPS pipeline. DCB identifies if an instruction is dependent on the instructions executing in the pipeline. The other purpose of the DCB is to ascertain that the required input operands are ready and that it is safe for the instruction to move into the execution stage.

a. What is the logic is to be implemented in the DCB to detect if an instruction in the decode stage is dependent on the instructions in the EX and MEM stage? (Use symbols similar to $I_{D\text{-}Rs}$, : $R_s$ field of the Instruction in the Decode stage. Psuedocode style answer is enough)

if(($I_{D\text{-}Rs}$== $I_{E\text{-}Rd}$)|| ($I_{D\text{-}Rt}$== $I_{E\text{-}Rd}$)|| ($I_{D\text{-}Rs}$== $I_{M\text{-}Rd}$)|| ($I_{D\text{-}Rt}$== $I_{M\text{-}Rd}$))
dependence - stall
else
no dependence

b. Is it necessary to check if the instruction in the decode stage is dependent on the instruction in the WB stage? Justify.

Yes. If not, the new value of register will be written into the register file in the write-back stage after the old value of register is read out by instruction in the decode stage.

3. What is the logic to be implemented inside the Forwarding Unit?

if($I_{D\text{-}Rs}$== $I_{E\text{-}Rd}$) forward Ex->Ex
if($I_{D\text{-}Rt}$== $I_{E\text{-}Rd}$) forward Ex->Ex
if($I_{D\text{-}Rs}$== $I_{M\text{-}Rd}$) forward MEM->Ex
if($I_{D\text{-}Rt}$== $I_{M\text{-}Rd}$) forward MEM->Ex

4. Identify all the dependences in the following snippet of code. Classify them. FX and RX are floating point and Integer registers.

```
i1: L.D      F4, 0(R0)
i2: MULT.D   F2, F0, F2
i3: DIVD     F8,F4,F2
i4: L.D      F4, 0(R1)
i5: ADDD     F6, F0, F4
i6: SUBD     F8, F8, F6
i7: SD       F8, 0(Ry)
```

RAW on F4 from i1 to i3
RAW on F2 from i2 to i3
RAW on F8 from i3 to i6
RAW on F4 from i4 to i5
RAW on F6 from i5 to i6
RAW on F8 from i6 to i7
WAR on F4 from i3 to i4
WAW on F4 from i1 to i4
WAW on F8 from i3 to i6

5. Write the pipeline diagrams for the code sequence.
a. without forwarding. CPI?

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw $t1,0($t0) | IF | ID | EX | MEM | WB | | | | | | | | | | | | | | |
| lw $t2, 4($t0) | | IF | ID | EX | MEM | WB | | | | | | | | | | | | | |
| add $t3, $t1, $t2 | | | IF | | | ID | EX | MEM | WB | | | | | | | | | | |
| sw $t3, 12($t0) | | | | | | IF | | | ID | EX | MEM | WB | | | | | | | |
| add $t4, $t9, $t7 | | | | | | | | | IF | ID | EX | MEM | WB | | | | | | |
| add $t5, $t1, $t4 | | | | | | | | | | IF | | | ID | EX | MEM | WB | | | |
| sub $t7, $t5, $t3 | | | | | | | | | | | | | IF | | | ID | EX | MEM | WB |

CPI=2.14

b. with forwarding. CPI?

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw $t1, 0($t0) | IF | ID | EX | MEM | WB | | | | | | | |
| lw $t2, 4($t0) | | IF | ID | EX | MEM | WB | | | | | | |
| add $t3, $t1, $t2 | | | IF | | ID | EX | MEM | WB | | | | |
| sw $t3, 12($t0) | | | | | IF | ID | EX | MEM | WB | | | |
| add $t4, $t9, $t7 | | | | | | IF | ID | EX | MEM | WB | | |
| add $t5, $t1, $t4 | | | | | | | IF | ID | EX | MEM | WB | |
| sub $t7, $t5, $t3 | | | | | | | | IF | ID | EX | MEM | WB |

CPI= 1.14

c. reorder the instructions from (b) to eliminate stalls. draw the pipeline diagram. CPI?

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| lw $t1, 0($t0) | IF | ID | EX | MEM | WB | | | | | | |
| lw $t2, 4($t0) | | IF | ID | EX | MEM | WB | | | | | |
| add $t4, $t9, $t7 | | | IF | ID | EX | MEM | WB | | | | |
| add $t3, $t1, $t2 | | | | IF | ID | EX | MEM | WB | | | |
| sw $t3, 12($t0) | | | | | IF | ID | EX | MEM | WB | | |
| add $t5, $t1, $t4 | | | | | | IF | ID | EX | MEM | WB | |
| sub $t7, $t5, $t3 | | | | | | | IF | ID | EX | MEM | WB |

CPI=7/7=1

<table>
<tr><td colspan="4">6. We examine how data dependences affect execution in the basic 5-stage pipeline. Use the following sequence of instructions. Also, assume the following cycle times for each of the options related to forwarding:</td><td rowspan="3">or r1, r2, r3<br>or r2, r1, r4<br>or r1, r1, r2</td></tr>
<tr><td></td><td>Without forwarding</td><td>With full forwarding</td><td>With ALU-ALU forwarding only</td></tr>
<tr><td></td><td>250ps</td><td>300ps</td><td>290ps</td></tr>
</table>

a. Indicate dependences and their type.

| Instruction sequence | Dependences |
|---|---|
| i1: or r1, r2, r3 | RAW on r1 from i1 to i2 and i3 |
| i2: or r2, r1, r4 | RAW on r2 from i2 to i3 |
| i3: or r1, r1, r2 | WAR on r2 from i1 to i2 |
| | WAR on r1 from i2 to i3 |
| | WAW on r1 from i1 to i3 |

b. Assume there is no forwarding in this pipelined processor. Indicate hazards and add nop instructions to eliminate them.

```
or r1, r2, r3
nop
nop                    delay i2 to avoid raw hazard on r1 from i1
or r2, r1, r4
nop
nop                    delay i3 to avoid raw hazard on r2 from i2
or r1, r1, r2
```

c. Assume there is full forwarding. Indicate hazards and add NOP instructions to eliminate them.

```
or r1, r2, r3
or r2, r1, r4          no raw hazard on r1 from i1 (forwarded)
or r1, r1, r2          no raw hazard on r2 from i2 (forwarded)
```

d. What is the total execution time of this instruction sequence without forwarding and with full forwarding? What is the speedup achieved by adding full forwarding to a pipeline that had no forwarding?

| No forwarding | with forwarding | speedup due to forwarding |
|---|---|---|
| (7+4)*250ps=2750ps | 7*300ps=2100ps | 1.30 |

e. Add nop instructions to this code to eliminate hazards if there is ALU-ALU forwarding only (no forwarding from the MEM to the EX stage).

```
or r1, r2, r3
or r2, r1, r4          ALU-ALU forwarding of r1 from i1
or r1, r1, r2          ALU-ALU forwarding of r2 from i2
```

f. What is the total execution time of this instruction sequence with only ALU-ALU forwarding? What is the speedup overall no-forwarding pipeline?

| No forwarding | With ALU-ALU forwarding only | Speedup with ALU-ALU forwarding |
|---|---|---|
| (7+4)*250ps=2750ps | 7*290ps=2030ps | 1.35 |