

CIS 351 Sample P2 Problem Solutions

Sun 19th Oct, 2025

P2: Data Hazards

- (a) Identify the 5 data hazards in the following code fragment. For each hazard, state whether it can be solved by forwarding or if stalls are necessary. Assume branches are resolved in the decode step.

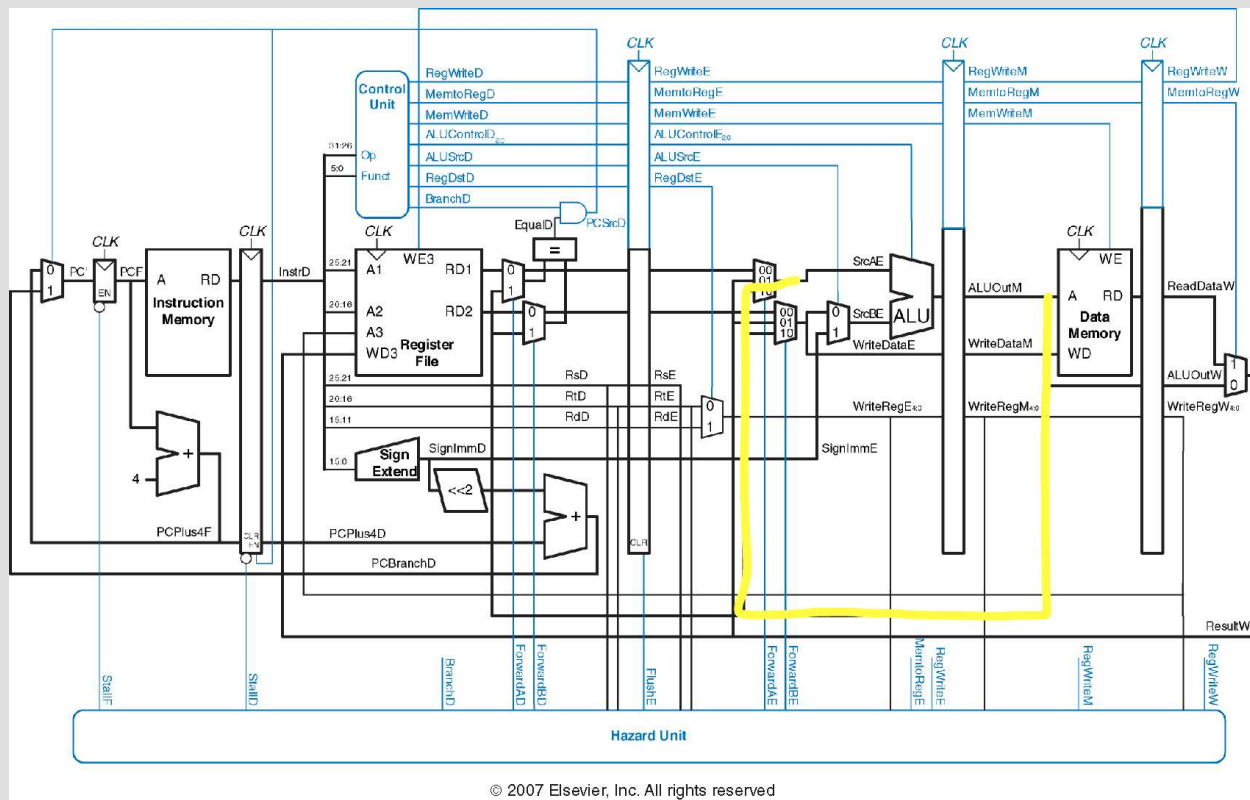
```
1. addi $v0, $zero, 0
2. sll  $a1, $a1, 2
3. add  $t1, $a0, $a1      # depends on 2. Forwarding works
4. lw   $t2, 0($t1)        # depends on 3. Forwarding works.
5. add  $v0, $v0, $t2      # depends on 4. Must stall.
6. addi $t1, $t1, 4        # depends on 3. Forwarding works.
7. slt  $t3, $v0, $zero    # depends on 5. Forwarding works.
```

- (b) Draw a pipeline diagram showing the execution of the code above on a 5-stage pipeline with no data or register forwarding.

```
addi $v0, $zero, 0  IF ID EX ME WB
sll  $a1, $a1, 2    IF ID EX ME WB
add  $t1, $a0, $a1  IF .. .. ID EX ME WB
lw   $t2, 0($t1)    IF .. .. ID EX ME WB
add  $v0, $v0, $t2  IF .. .. ID EX ME WB
addi $t1, $t1, 4    IF ID EX ME WB
slt  $t3, $v0, $zero IF .. .. ID EX ME WB
```

(c) Highlight the forwarding path used by \$t1 between the first and second instruction below.

```
addi $t1, $zero, 15
add $t5, $t1, $t7
```



(d) Explain how the use of branch delay slots allows the pipelined processor to “handle” control hazards without adding extra hardware. If the hardware isn’t “handling” the hazard, what is?

The assembler / programmer is handling the control hazard explicitly.

- (e) Draw a pipeline diagram showing the execution of the code below on a 5-stage pipeline that has data forwarding, but handles control hazards by stalling.

```
    addi $t0, $zero, 15
    addi $t1, $zero, 15
    beq $t0, $t1, target
    sub $t1, $t2, $t3
    add $t4, $t5, $t6
    lw $t7, 0($a0)
target: add $v0, $zero, 100
```

1. addi \$t0, \$zero, 15	IF ID EX ME WB
2. addi \$t1, \$zero, 15	IF ID EX ME WB
3. beq \$t0, \$t1, target	IF .. ID EX ME WB # See note below.
4. sub \$t1, \$t2, \$t3	
5. add \$t4, \$t5, \$t6	
6. lw \$t7, 0(\$a0)	
target: add \$v0, \$zero, 100	IF ID EX ME WB

The beq stalls because the computation of \$t1 is still in the EX stage when the beq first attempts to compare the two registers during the ID stage.