Discussion 11b: Stalling, Forwarding and Flushing

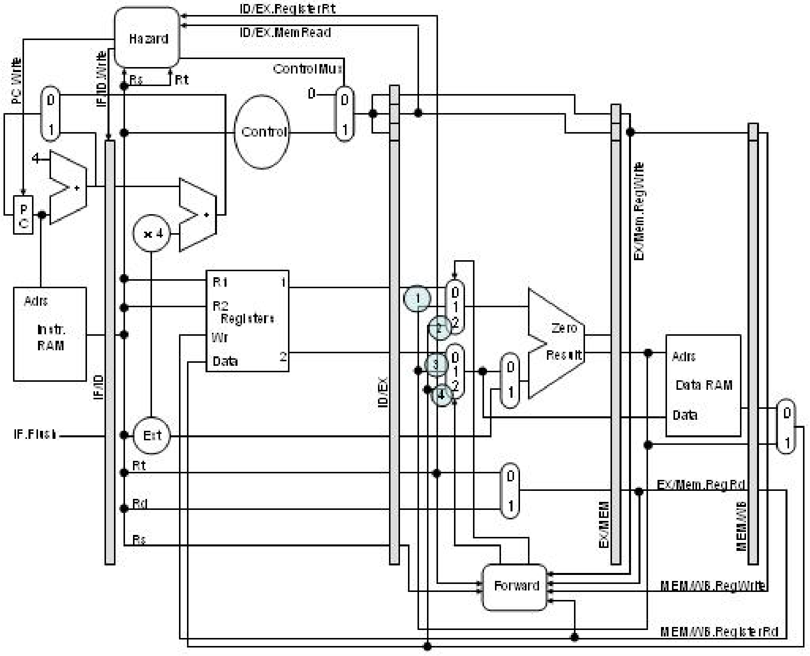
1 Stalling vs. Forwarding

Suppose we have the following chunk of code containing only R-type instructions.

```
add $8,$5,$5
add $2,$5,$8
sub $3,$8,$4
add $2,$2,$3
```

1. Identify the hazards involved (draw the arrows between *dependencies* that cause da hazards).

2. Figure below shows the pipelined datapath with four forwarding inputs. For each depe dency identified above specify which numbered forwarding path is used.



3. Fill out the following pipeline diagram, for the case when forwarding is implemented.

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
`add $8,$5,$5`															
`add $2,$5,$8`															
`sub $3,$8,$4`															
`add $2,$2,$3`															

4. Fill in the pipeline diagram below, for the case when forwarding is **not** implemented.

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
`add $8,$5,$5`															
`add $2,$5,$8`															
`sub $3,$8,$4`															
`add $2,$2,$3`															

2 Rewriting Code to Avoid Stalls

(a) Identify all the data hazards in the below code that can't be resolved by forwarding.

```
lw    $t0, 0($a0)
addi  $t0, $t0, 1
sw    $t0, 0($a0)
addi  $a0, $a0, 4
```

(b) Rewrite the code to eliminate stalls on the 5-stage pipeline with full forwarding.

3 Jumps and Branches

(a) Assume the target of a `j` instruction (unconditional jump) can be computed in the ID stage of the 5-stage MIPS pipeline. How many instructions must be flushed each time a j instruction is executed? Draw a pipeline diagram to support your answer.

The following code is run on the 5-stage MIPS pipeline with full forwarding. Branch targets and decisions are resolved in the **EX stage**, and branches are always predicted **not taken**.

```
  # a0 = head node of list
  li   $v0, 0           # sum = 0
loop:
  beq  $a0, $0, done    # while a0 != NULL
  lw   $t0, 0($a0)      # t0 = a0->data
  add  $v0, $v0, $t0    # sum += t0
  lw   $a0, 4($a0)      # a0 = a0->next
  j    loop
done:
  sw   $v0, 16($sp)
```

(b) If the linked list has a length n, how many cycles will the above code take to execute (including the `sw`)?

(c) Given that $n > 0$, would it be beneficial to convert this into a `do-while` style loop?