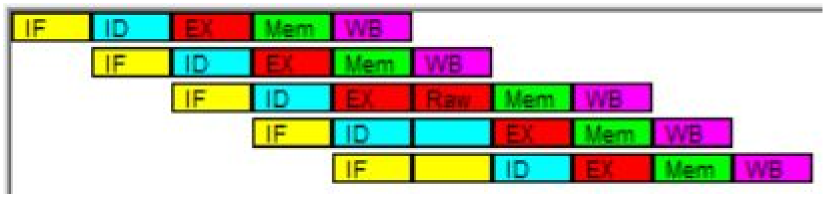
CS341 Assignment 7(lab 4)

Team:
Guttu Sai Abhishek 180050036
Hrithik Vikas Samala 180050038
Mulinti Shaik Wajid 180050063
Sai Phanindra Ramasahayam 180050084
Sanapathi Sumanth Balaji 180050091

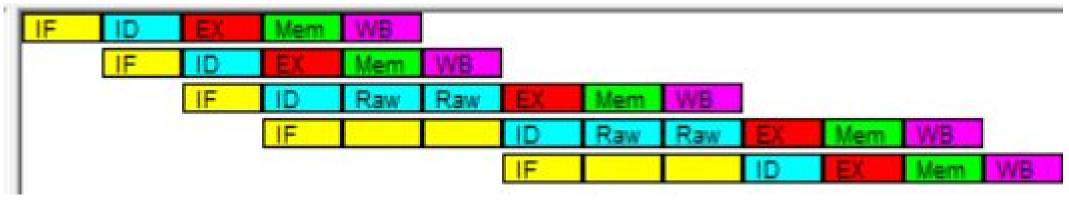
Question 1

```
.data
.text
main:
      lw r1, 0($sp)
      lw r2, 4($sp)
      dadd r3, r1, r2
      dadd r4, r1, r3
      halt
```

With forwarding: 10 cycles, 5 instructions, 2 CPI



Without forwarding: 13 cycles, 5 instructions, 2.60 CPI



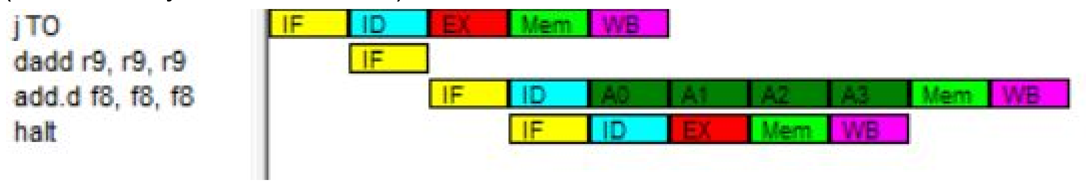
Question 2

Code 1:

```
.text
main:
      j TO
      dadd r9, r9, r9
      dadd r10, r10, r10
      dadd r11, r11, r11
      dadd r12, r12, r12
```

```
     TO:
     add.d f8, f8, f8
     halt
```

(branch delay slot not enabled)



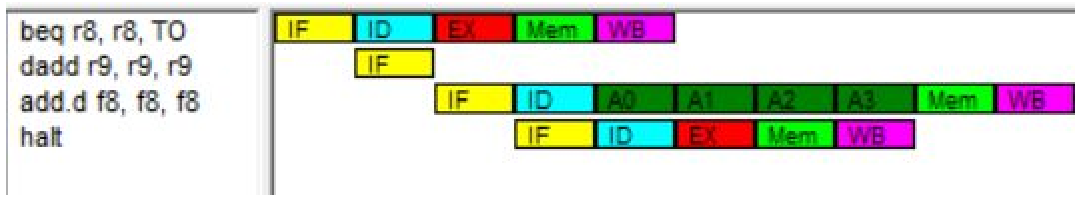
```
j TO               IF  ID  EX  Mem WB
dadd r9, r9, r9        IF
add.d f8, f8, f8           IF  ID  A0  A1  A2  A3  Mem WB
halt                           IF  ID  EX  Mem WB
```

Code 2:

```
.text
main:
     beq r8, r8, TO
     dadd r9, r9, r9
     dadd r10, r10, r10
     dadd r11, r11, r11
     dadd r12, r12, r12

     TO:
     add.d f8, f8, f8
     halt
```

(branch delay slot not enabled)

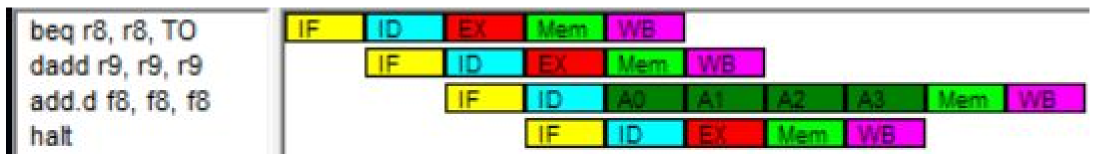


```
beq r8, r8, TO     IF  ID  EX  Mem WB
dadd r9, r9, r9        IF
add.d f8, f8, f8           IF  ID  A0  A1  A2  A3  Mem WB
halt                           IF  ID  EX  Mem WB
```

Code 3:

Same code as code 2

(branch delay slot enabled)



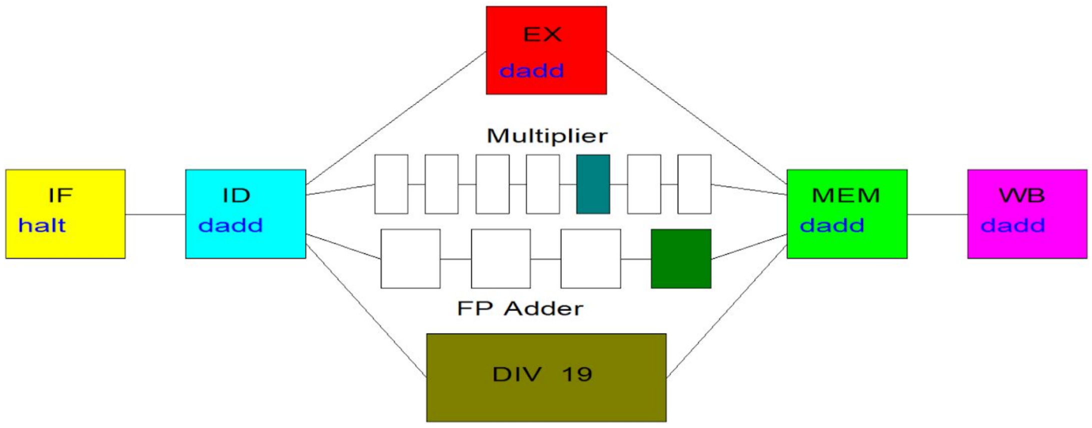
```
beq r8, r8, TO     IF  ID  EX  Mem WB
dadd r9, r9, r9        IF  ID  EX  Mem WB
add.d f8, f8, f8           IF  ID  A0  A1  A2  A3  Mem WB
halt                           IF  ID  EX  Mem WB
```

a) Jump instruction has no condition to check unlike conditional branch instructions. But it still has to calculate the target address. And from the pipeline diagram of code 1, the target address during jump instruction is calculated by the end of ID stage. If the target address was computed in IF stage, then there should be no bubble in dadd instruction i.e., target address is computed in the ID stage. For conditional branch in code 2, it can be seen that branch target address is computed by the end of the ID stage and since jump instruction itself computes target address in ID stage implies that conditional branch also computes branch target address in ID stage.
b) From the pipeline diagram of code 2, there is only one bubble during dadd instruction i.e., both branch condition and branch target address should be computed by the end of the ID stage. From part a, the target address is computed in the ID stage. Therefore the branch condition is evaluated by the end of ID stage i.e., branch condition is evaluated in the ID stage.
c) From pipeline diagram of code 2, one stall due to a taken branch without a branch delay slot.
d) From pipeline diagram of code 3, NO stall is taken due to a taken branch with a branch delay slot. It is the duty of the programmer/compiler to put an useful instruction just after the branch instruction when the branch delay slot is enabled.

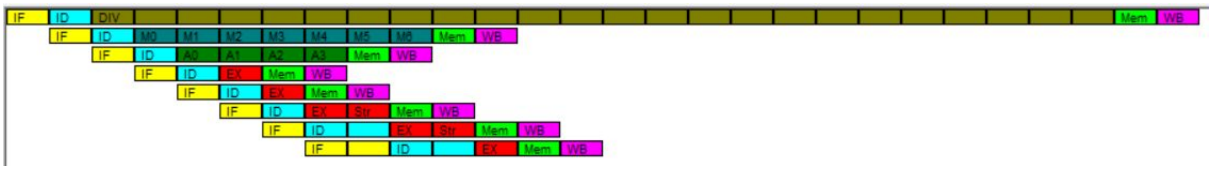
Question 3

```
.text
main:
  div.d f7, f5, f3
  mul.d f10, f4, f4
  add.d f1, f1, f1
  dadd r4, r5, r6
  dadd r7, r7, r7
  dadd r8, r8, r8
  dadd r9, r9, r9
  halt
```

All functional units are used in cycle number 8. The diagram is attached below



(with or without forwarding)

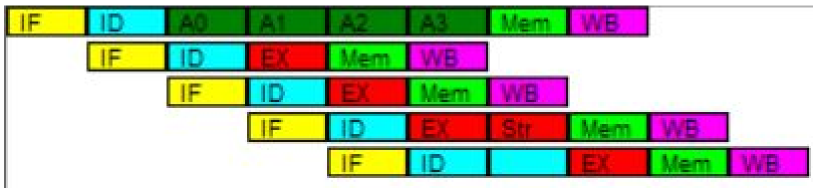


Question 4

```
.text
main:
  add.d f8, f8, f8
  dadd r4, r5, r6
  dadd r7, r7, r7
  dadd r8, r8, r8
  halt
```

The earlier instruction(the instruction which has appeared first) gets promoted to MEM stage. In the above example both 1st and 4th instructions try to access Data Memory at same time and add.d(1st instruction) gets promoted to MEM instead of fourth instruction(`dadd r8, r8, r8`)

(with or without forwarding)



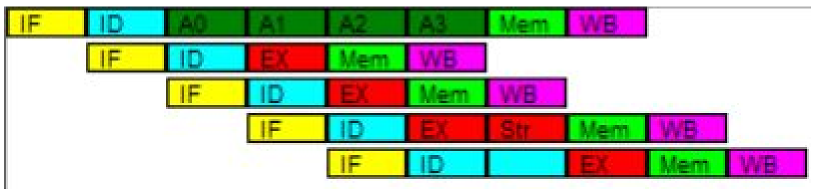
Question 5

<u>Structural hazard due to occupied Data Memory(at MEM stage)</u>

```
.text
main:
  add.d f8, f8, f8
  dadd r4, r5, r6
  dadd r7, r7, r7
  dadd r8, r8, r8
  halt
```

The fourth instruction can not proceed to the MEM stage because the first instruction is in MEM stage at cycle number 7, both first and fourth instruction completed EX stage at the same time and want to go to MEM stage in the next cycle but both can not go to MEM stage simultaneously and the fourth instruction is stalled. The component causing the structural hazard is Data memory.

(with or without forwarding)

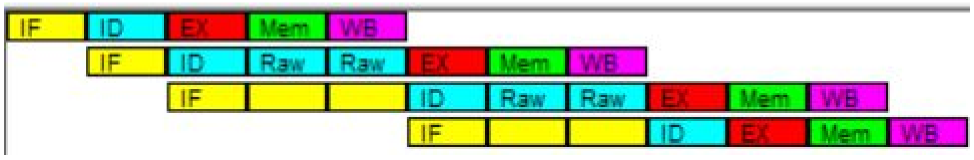


<u>Structural Hazard due to occupied Registers(during ID stage)</u>

```
.text
main:
      dadd r8, r9, r10
      lw r10, 100(r8)
      dadd r9, r10, r8
      halt
```

The 3rd instruction at 4th cycle is structurally stalled because of the RAW stalled ID stage in 2nd instruction. And the structural Hazard is due the still in use registers Datapath element by 2nd instruction and the same is needed by 3rd instruction.

(without forwarding)



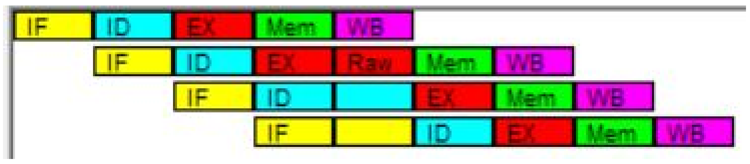
<u>Structural Hazard due to occupied ALU (during EX stage)</u>

```
.text
main:
      lw r5, 100(r5)
```

```
        dadd r6, r5, r6
        dadd r7, r7, r7
        halt
```

The 3rd instruction at 5th cycle is structurally stalled because of RAW stalled EX stage in 2nd instruction.And the structural Hazard is due the still in use ALU Datapath element by 2nd instruction and the same is needed by 3rd instruction.

(with forwarding)

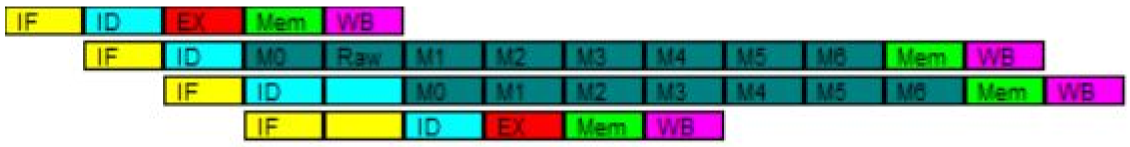


Structural Hazard due to occupied Multiplier

```
.text
main:
        l.d f5, 100(r5)
        mul.d f6, f5, f6
        mul.d f7, f7, f7
        halt
```

The 3rd instruction at 5th cycle is structurally stalled because of RAW stalled M(mult) stage in 2nd instruction.And the structural Hazard is due the still in use multiplier Datapath element by 2nd instruction and the same is needed by 3rd instruction.

(with forwarding)

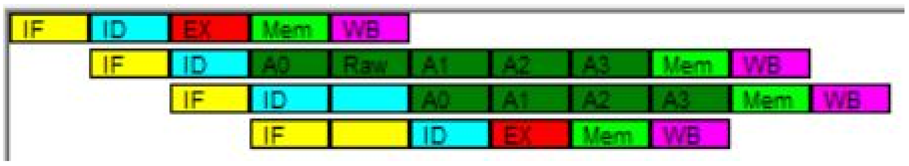


Structural Hazard due to occupied FP adder

```
.text
main:
        l.d f5, 100(r5)
        add.d f6, f5, f6
        add.d f7, f7, f7
        halt
```

The 3rd instruction at 5th cycle is structurally stalled because of RAW stalled A(FP adder) stage in 2nd instruction.And the structural Hazard is due the still in use FP adder Datapath element by 2nd instruction and the same is needed by 3rd instruction.

(with forwarding)

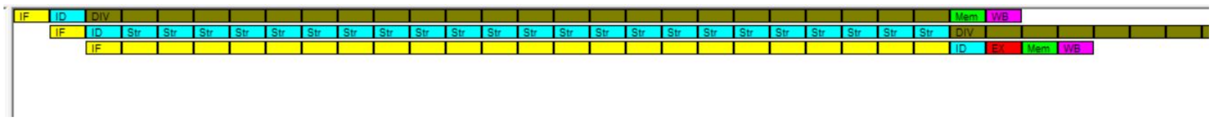


Structural Hazard due to occupied divider

```
.text
main:
      ddiv r8, r8, r8
      ddiv r7, r7, r7
      halt
```

The 2nd instruction at 4th cycle is structurally stalled because of DIV(divider) stage in 1st instruction.And the structural Hazard is due the still in use divider Datapath element by 1st instruction and the same is needed by 2nd instruction.

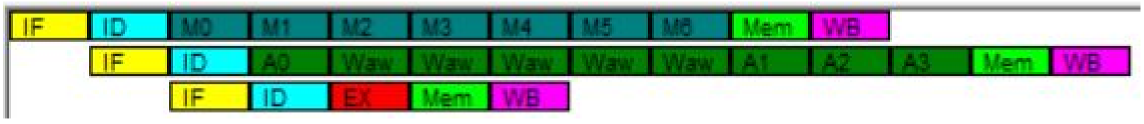
(without forwarding)



Question 6

```
.text
main:
      mul.d f4, f5, f6
      add.d f4, f7, f8
      halt
```

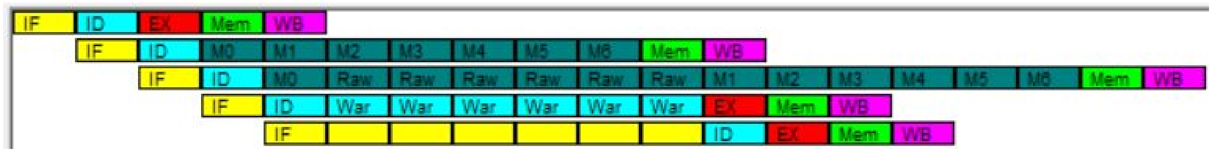
(without forwarding)


IF ID M0 M1 M2 M3 M4 M5 M6 Mem WB
IF ID A0 Waw Waw Waw Waw Waw A1 A2 A3 Mem WB
IF ID EX Mem WB

Question 7

```
.text
main:
      daddi r6, r9, 1
      dmulu r5, r9, r9
      dmulu r2, r5, r6
      daddi r5, r1, 1
      halt
```

(with forwarding)


IF ID EX Mem WB
IF ID M0 M1 M2 M3 M4 M5 M6 Mem WB
IF ID M0 Raw Raw Raw Raw Raw Raw M1 M2 M3 M4 M5 M6 Mem WB
IF ID War War War War War War EX Mem WB
IF ID EX Mem WB