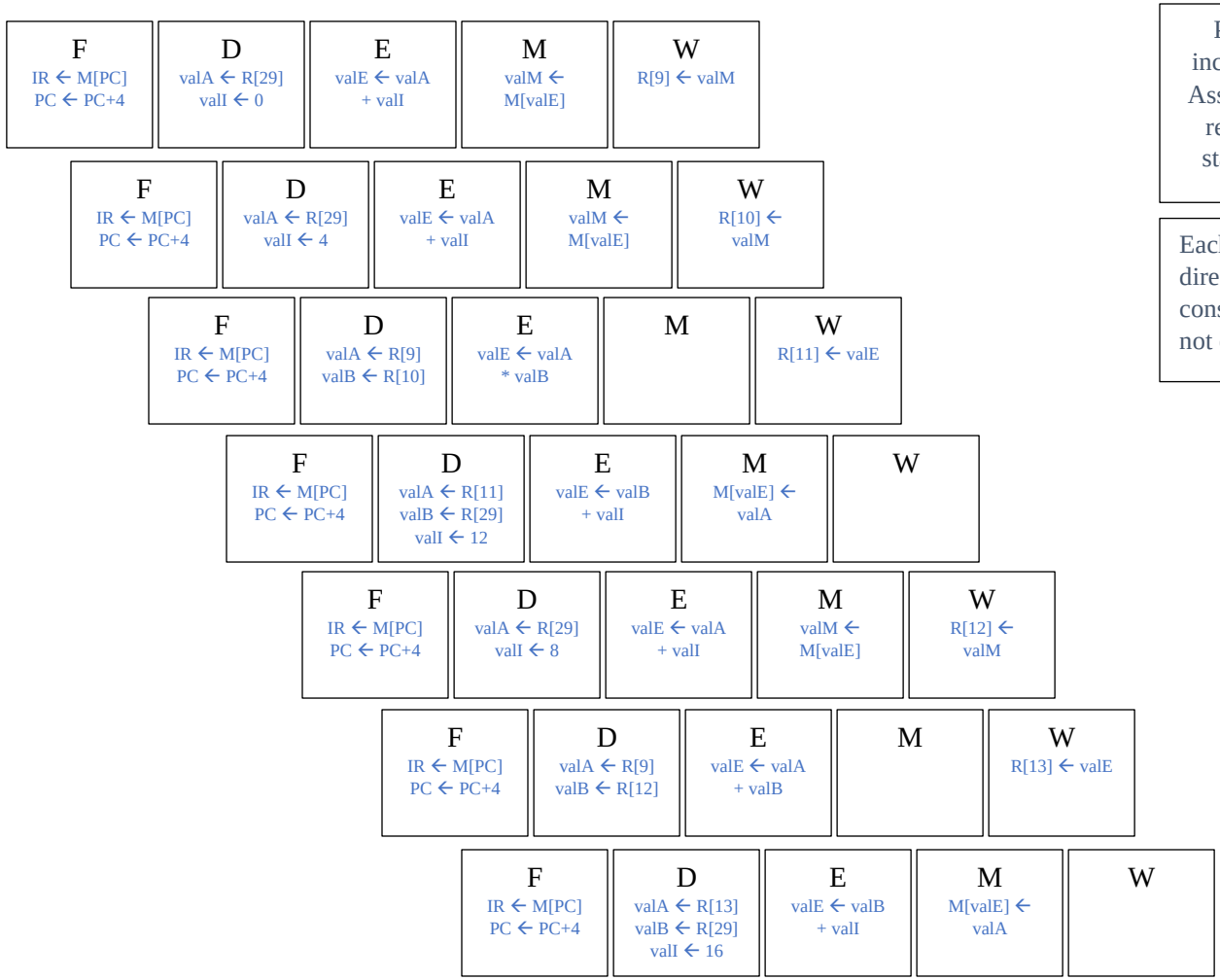
```
# Loading b from the stack
1. lw $t1, 0($sp)

# Loading e from the stack
2. lw $t2, 4($sp)

# $t3 = $t1 * $t2
3. mul $t3, $t1,$t2

# Storing a onto the stack
4. sw $t3, 12($sp)

# Loading f from the stack
5. lw $t4, 8($sp)

# $t5 = $t1 + $t4
6. add $t5, $t1,$t4

# Storing c onto the stack
7. sw $t5, 16($sp)
```



Pipeline registers not included in the diagrams. Assume there are pipeline registers between each stage and after W stage

Each clock cycle is not directly above the other to conserve space, this does not change functionality

C Code:

```
a = b * e;
c = b + f;
```

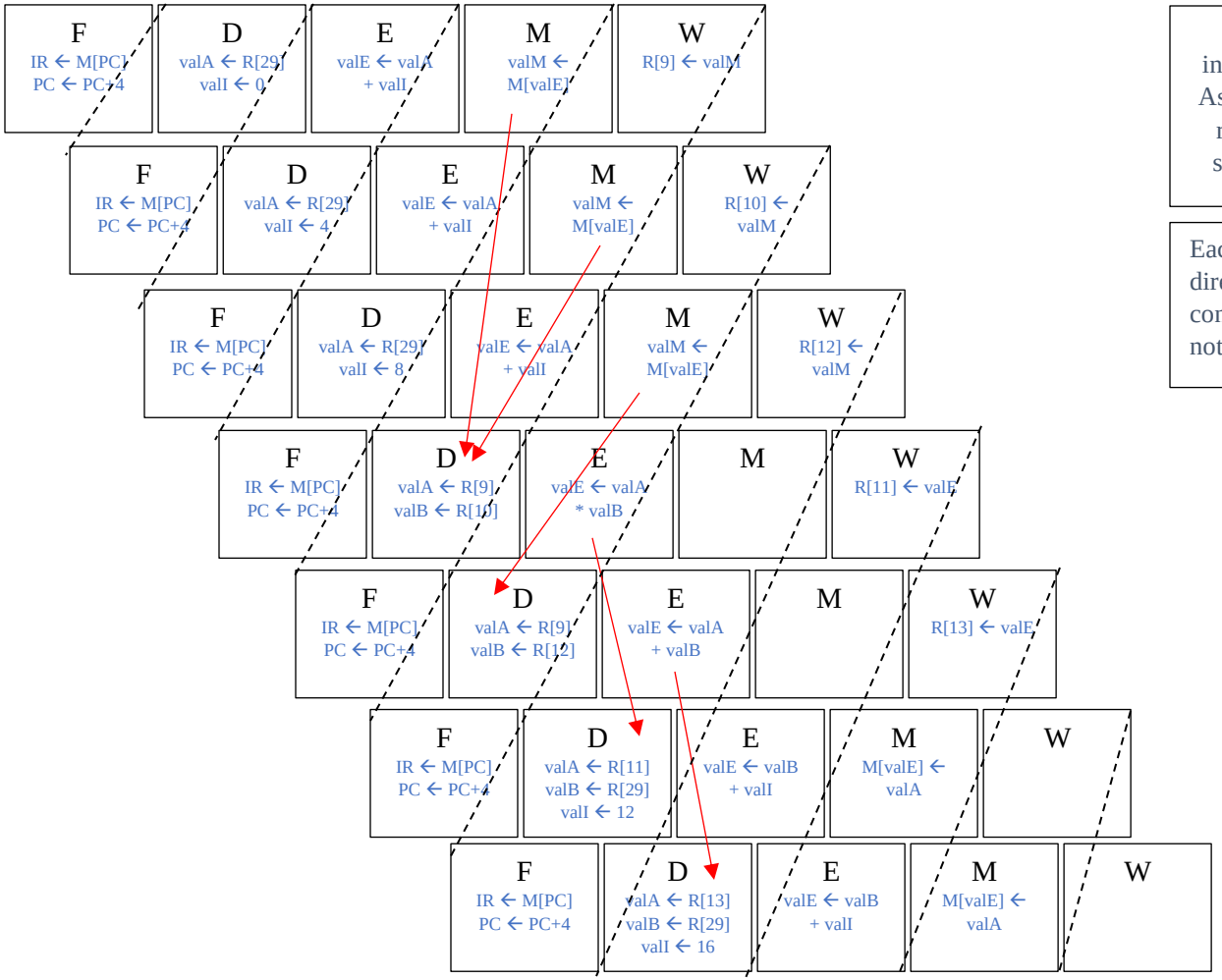
---

2. There are multiple data hazards within the MIPS machine instructions

Data Hazards:

- Instruction 3 depends on instruction 1 and 2 for $t1, $t2
- Instruction 4 requires instruction 3 to execute to get $t3
- Instruction 6 depends on instruction 1 and 5 for $t1, $t4
- Instruction 7 requires instruction 6 to execute to get $t5

```
# Loading b from the stack
lw $t1, 0($sp)

# Loading e from the stack
lw $t2, 4($sp)

# Loading f from the stack
lw $t4, 8($sp)

# $t3 = $t1 * $t2
mul $t3, $t1,$t2

# $t5 = $t1 + $t4
add $t5, $t1,$t4

# Storing a onto the stack
sw $t3, 12($sp)

# Storing c onto the stack
sw $t5, 16($sp)
```



Pipeline registers not included in the diagrams. Assume there are pipeline registers between each stage and after W stage

Each clock cycle is not directly above the other to conserve space, this does not change functionality

---

- The red lines symbolize data forwarding
  - Instruction 1 and 2 are forwarded to instruction 4
  - Instruction 3 is forwarded to instruction 5
  - Instruction 4 is forwarded to instruction 6
  - Instruction 5 is forwarded to instruction 7
- t1 has been written to by the time we get to the decode stage of the *add* instruction so data forwarding of $t1 is not required