Load-Use Hazard Detection

- Check when using instruction is decoded in ID stage
- ALU operand register numbers in ID stage are given by
  - IF/ID.RegisterRs1, IF/ID.RegisterRs2
- Load-use hazard when
  - ID/EX.MemRead and
    ((ID/EX.RegisterRd = IF/ID.RegisterRs1) or
    (ID/EX.RegisterRd = IF/ID.RegisterRs1))
- If detected, stall and insert bubble

How to Stall the Pipeline

- Force control values in ID/EX register to 0
  - EX, MEM and WB do `nop` (no-operation)
- Prevent update of PC and IF/ID register
  - Using instruction is decoded again
  - Following instruction is fetched again
  - 1-cycle stall allows MEM to read data for `ld`
    - Can subsequently forward to EX stage

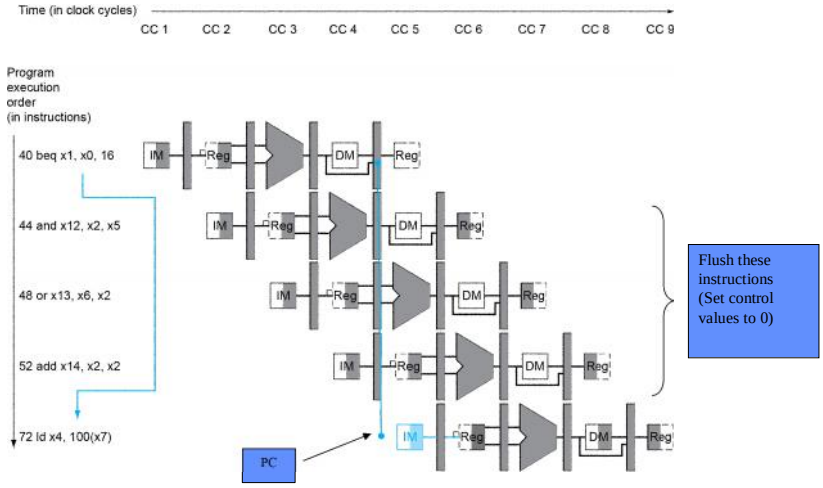
Stalls and Performance

The BIG Picture

- Stalls reduce performance
  - But are required to get correct results
- Compiler can arrange code to avoid hazards and stalls
  - Requires knowledge of the pipeline structure

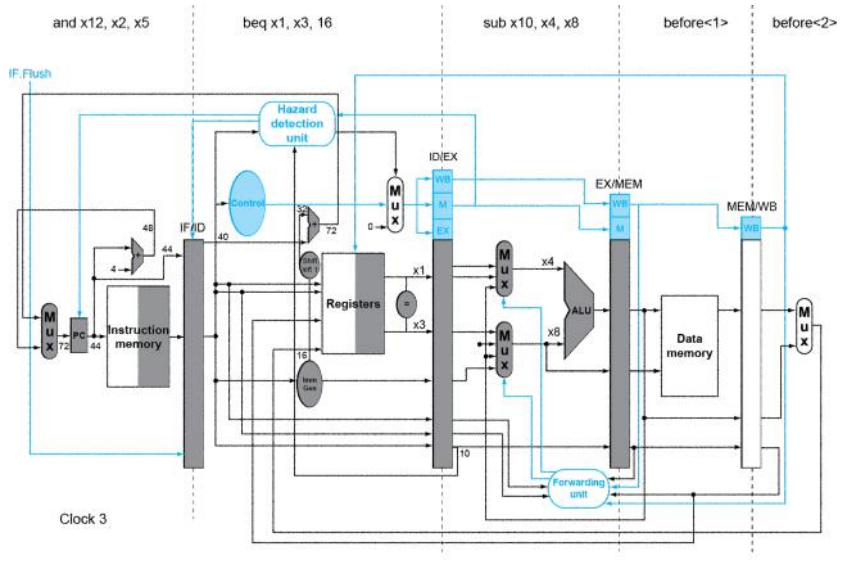
Branch Hazards

- If branch outcome determined in MEM



Control Hazards--How to handle?

- One software scheme: branch delay slot
  - Compiler inserts an instruction directly after a branch instruction that is independent of the branch inst.
    - No mater the branch is taken or not, that instruction should always be executed.
- One optimization
  - move branch addr calculation and condition test to ID stage
    - The lost 3 cycles is cut to 1.



Flushing instructions

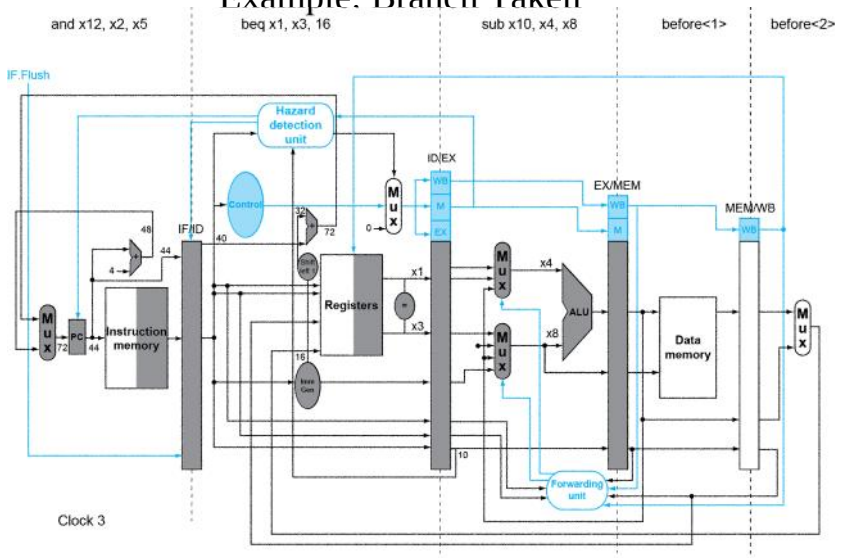
- Move branch addr. calculation/condition testing to stage 2 (ID)
- Condition testing: use 32 XOR gates to compare bitwisely
  - delay: negligible
- IF.Flush==branch control line in single cycle datapath
- $F = Zero \bullet IF.Flush$ is used to control write of PC & IF.ID buffer
- If F=1, take branch by doing two operations
  - zero IF.ID buffer (flushing)
  - write PC with branch address
- If branch taken, lose one cycle
- CPI of `beq`

Reducing Branch Delay

- Example: branch taken

```
36:  sub  x10, x4, x8
40:  beq  x1,  x3, 16  // PC-relative branch
                       // to 40+16*2=72
44:  and  x12, x2, x5
48:  orr  x13, x2, x6
52:  add  x14, x4, x2
56:  sub  x15, x6, x7
     ...
72:  ld   x4, 50(x7)
```

Example: Branch Taken



Example: Branch Taken