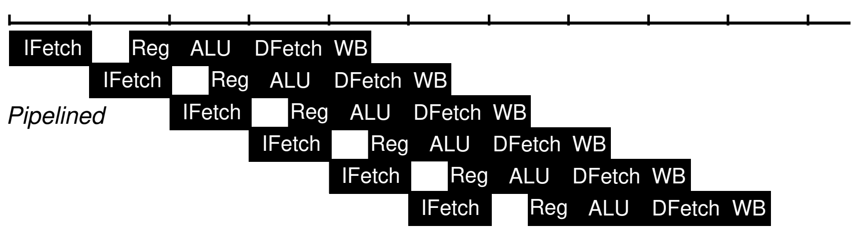
Hazard County

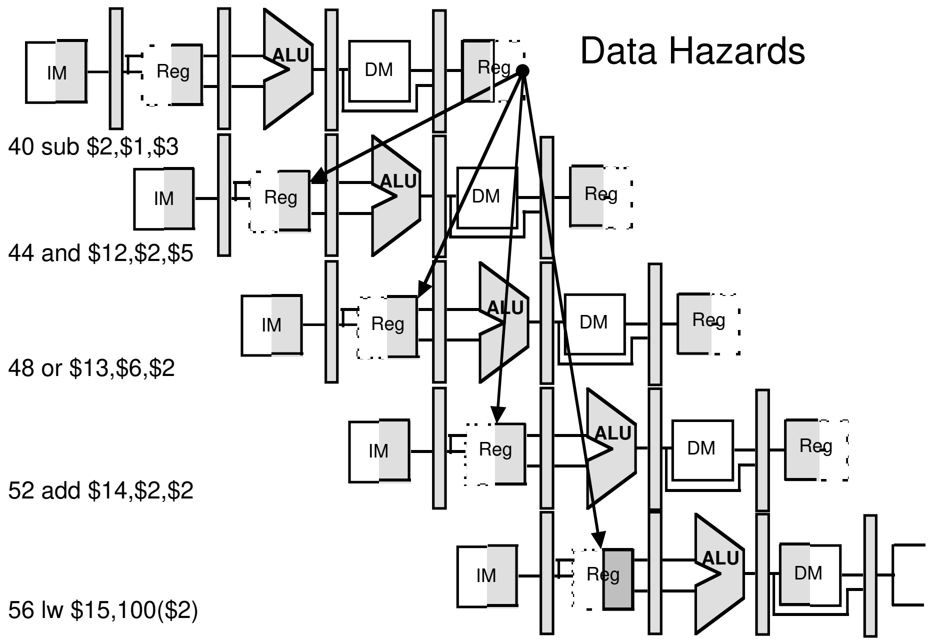
Pipelining is a good idea, but to make it work, several kinds of hazards have to be avoided, and interrupts must be handled.

Pipelining

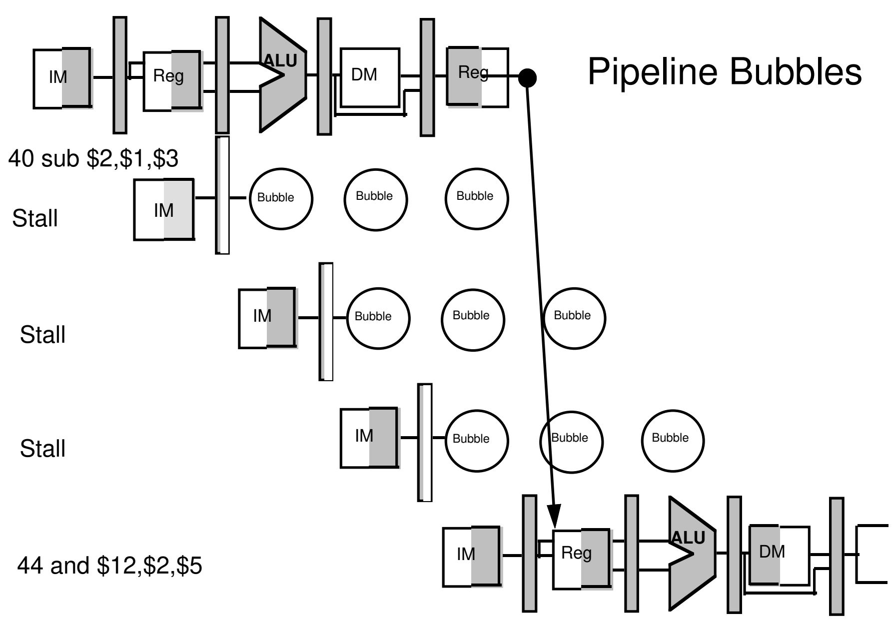
- Recall that pipelining initiates an operation on each clock cycle
- The first design goal is to assure that there are no conflicts in the use of the hardware



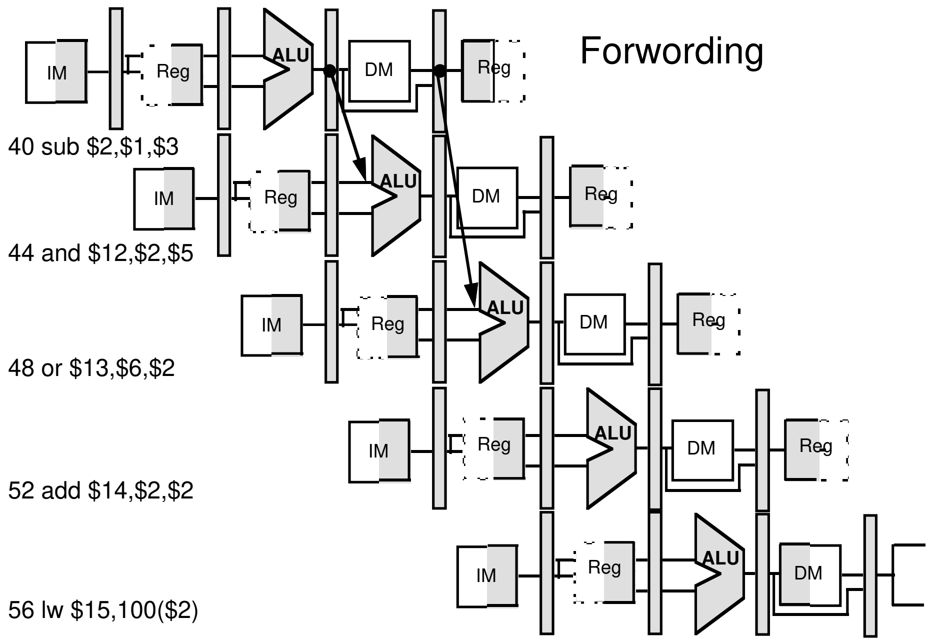
Data Hazards



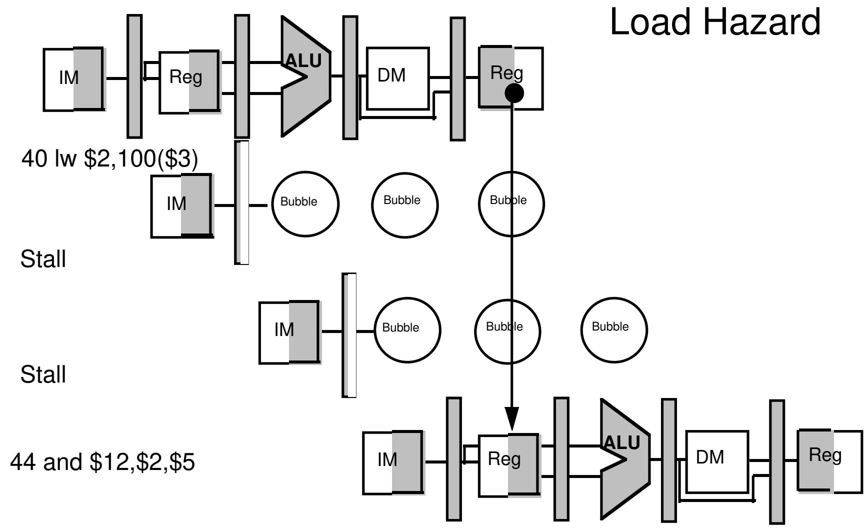
Pipeline Bubbles



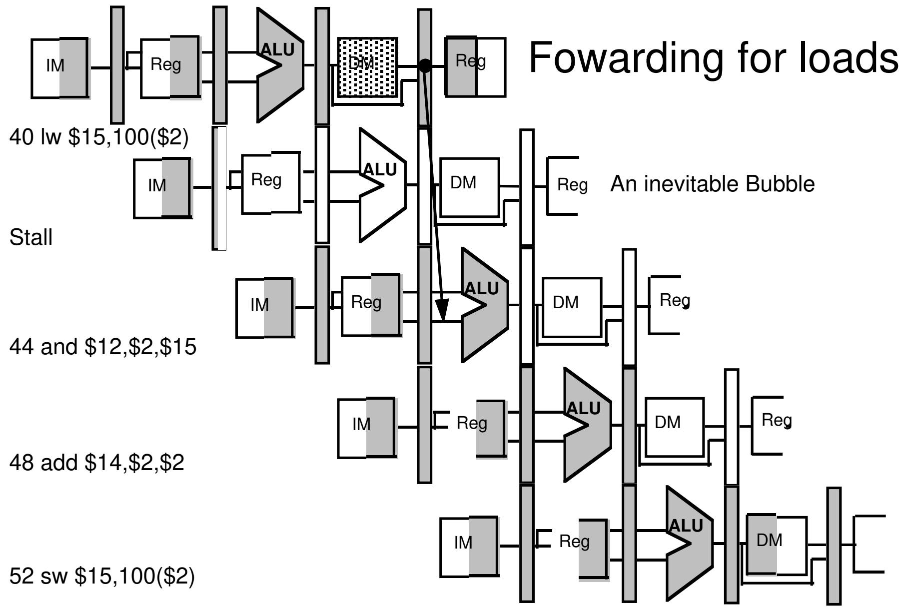
Forwording



Load Hazard



Fowarding for loads



Detecting Data Hazards

EX Hazard:

ID/EX.RegWrite and
((ID/EX.RegDst=0 and ID/EX.WriteRegisterRt=IF/ID.ReadRegister1)or
(ID/EX.RegDst=1 and ID/EX.WriteRegisterRd=IF/ID.ReadRegister1)or
(ID/EX.RegDst=0 and ID/EX.WriteRegisterRt=IF/ID.ReadRegister2)or
(ID/EX.RegDst=1 and ID/EX.WriteRegisterRd=IF/ID.ReadRegister2))

MEM Hazard:

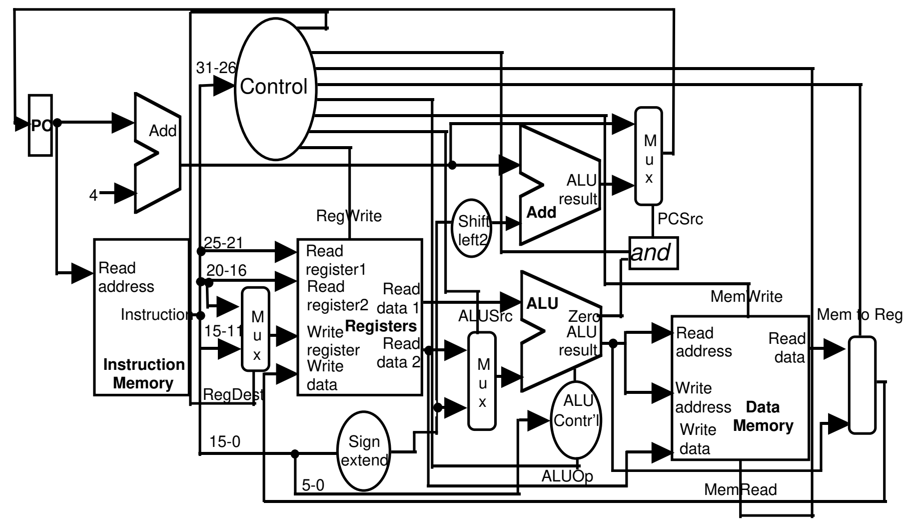
EX/MEM.RegWrite and
((EX/MEM.WriteRegister=IF/ID.ReadRegister1)or
(EX/MEM.WriteRegister=IF/ID.ReadRegister2))

WB Hazard:

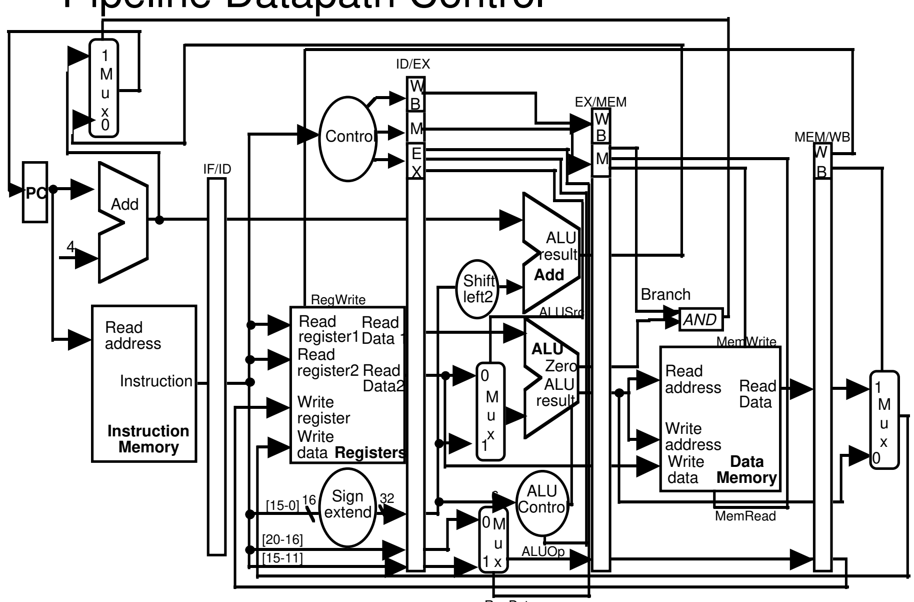
MEM/WB.RegWrite and
((MEM/WB.WriteRegister=IF/ID.ReadRegister1)or
(MEM/WB.WriteRegister=IF/ID.ReadRegister2))

Datapath Control

Control logic is generated from the opcode bits



Pipeline Datapath Control



Load Hazards

ID/EX.RegWrite and (ID/EX.RegDst = 0) and
((ID/EX.WriteRegisterRt = IF/ID.Read.Register1) or
(ID/EX.WriteRegisterRt = IF/ID.ReadRegister2))

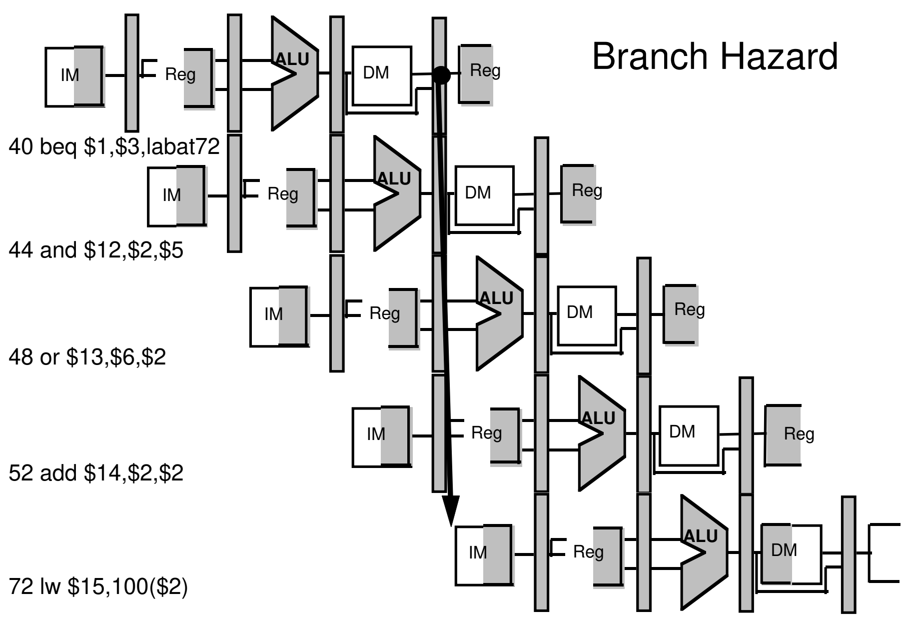
Consider the sequence to swap stack items

```
lw   $10, 0($29)  -- Load top item
lw   $11, 4($29)  -- Load second item
sw   $11, 0($29)  -- Replace top item
sw   $10, 4($29)  -- Replace second item
```

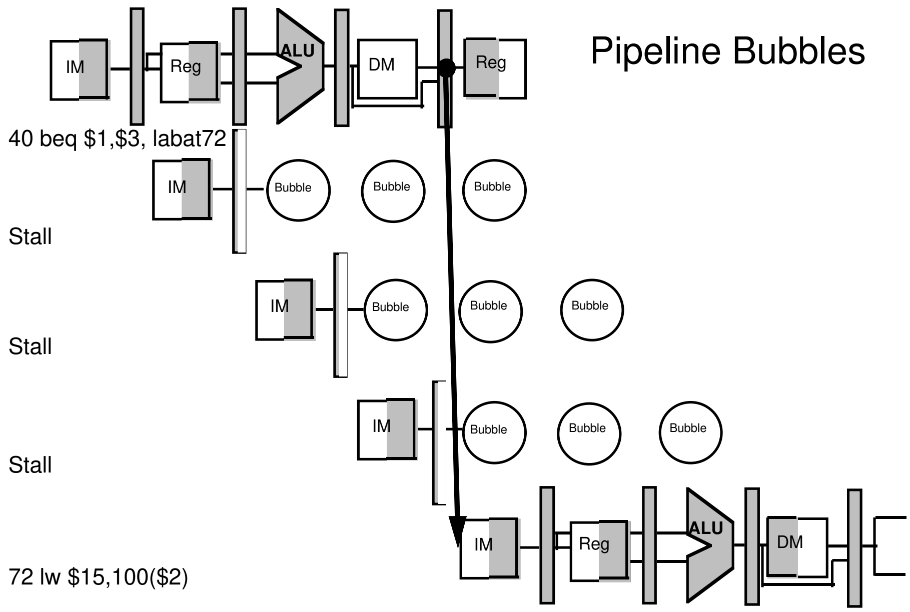
Which can be replace by

```
lw   $10, 0($29)  -- Load top item
lw   $11, 4($29)  -- Load second item
sw   $10, 4($29)  -- Replace second item
sw   $11, 0($29)  -- Replace top item
```

Branch Hazard



Pipeline Bubbles



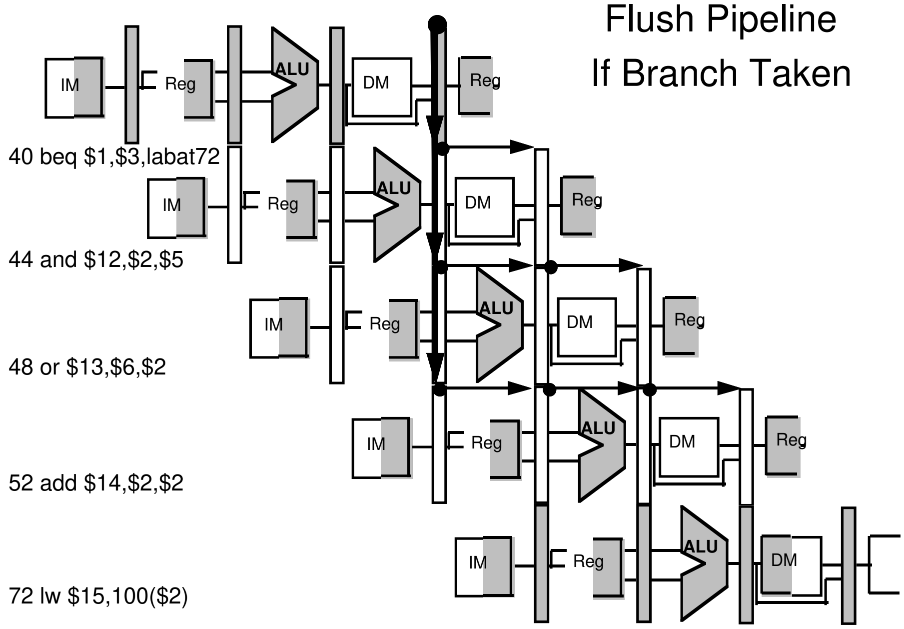
Branch Optimizations

If the compiler is assigned the task of filling in no-ops after a branch, then it can possibly fill in the slots with useful instructions.

Branch not taken

- Continue to execute instructions in order
- When the branch is determined in MEM, the instructions in the IF, ID and EX stages have to be flushed.
  - IF: Zero the instruction in the IF/ID pipeline register.
  - ID: Zero control signals like a stall.
  - EX: Zero control signals like a stall.

Flush Pipeline If Branch Taken



Exceptions

In pipelined datapaths, exceptions must flush instructions already in the pipe to avoid trashing the state, e.g. register contents, that caused the exception.

```
addi $8, $8, 1024 #Overflow except must save $8
```

Exception handling requires two steps:

- Use the same flush mechanism as the "taken" branches, to neutralize the affects of all instructions in the pipe.
- Load the exception address into the PC.

Imprecise interrupts.

Interrupts, e.g. from I/O devices, can be handled "when convenient," though continuing to process instructions has its risks

Multiple interrupts could occur simultaneously -- prioritize