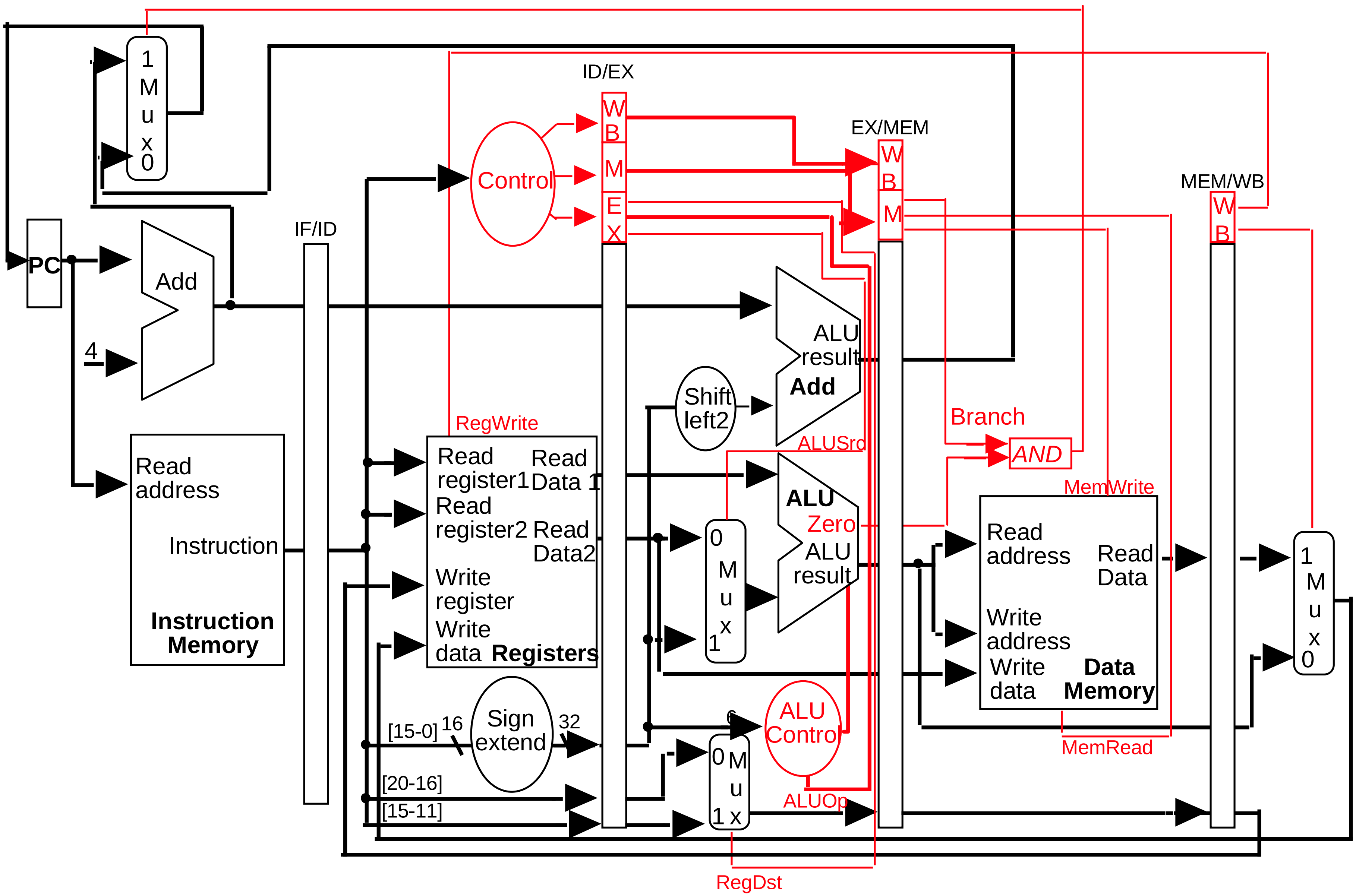
Branch Hazards

Data hazards can be helped by forwarding, but branch hazards are not so easily resolved.

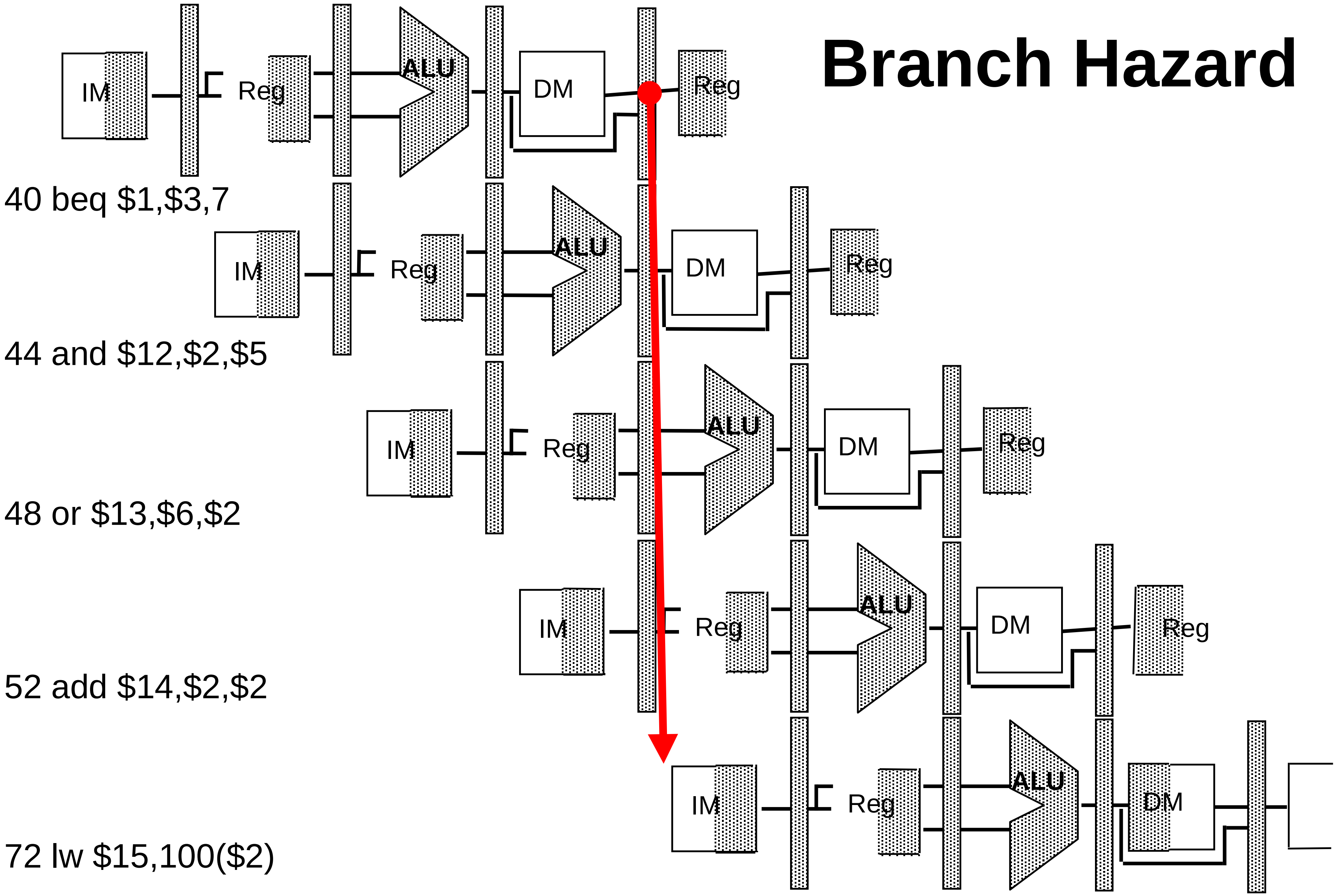
Recall that the branch decision is made in the MEM stage, which is late in the pipeline.

Two choices: reduce the branch delay, or attempt to predict branch results

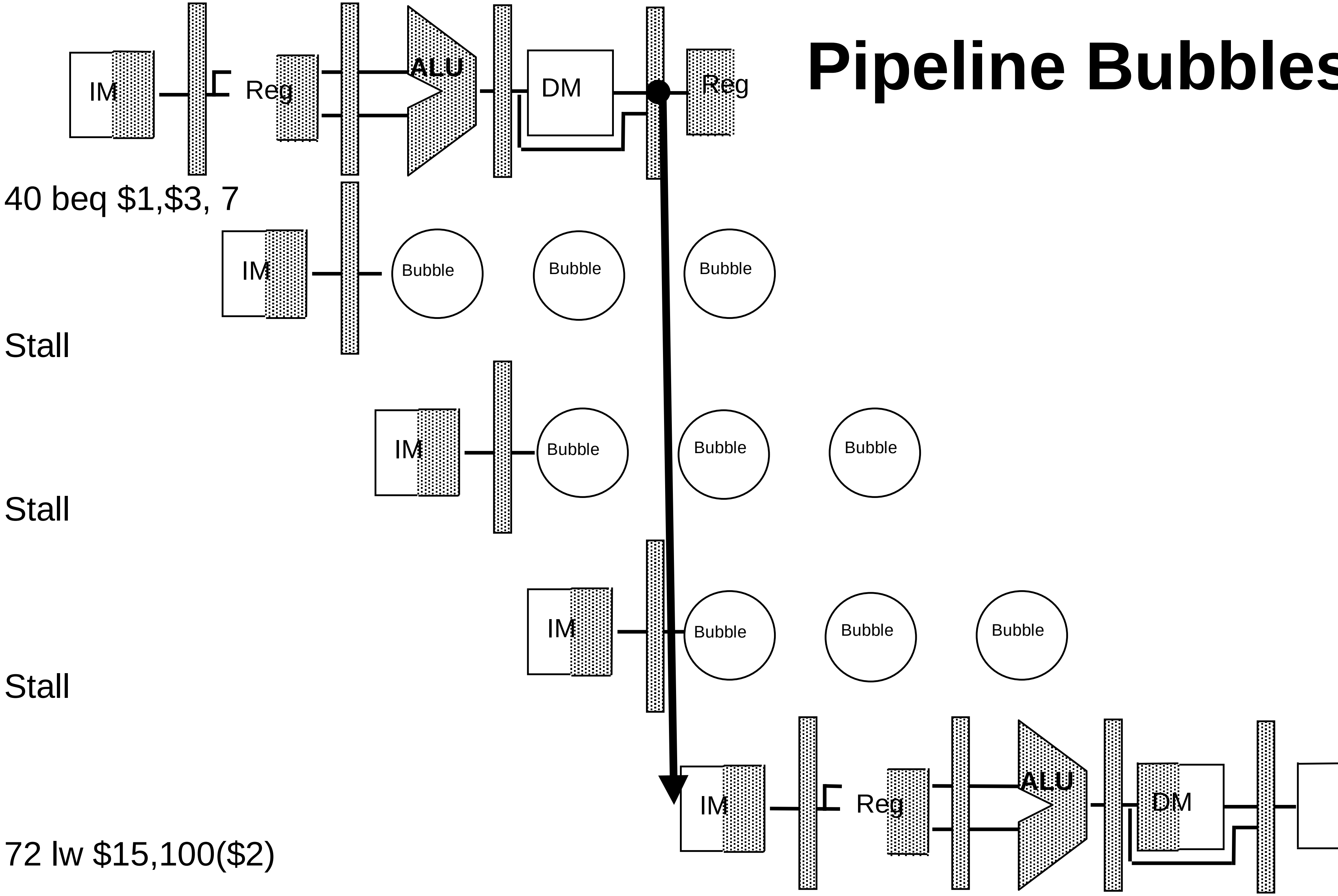
Branching determined at MEM



Branch Hazard



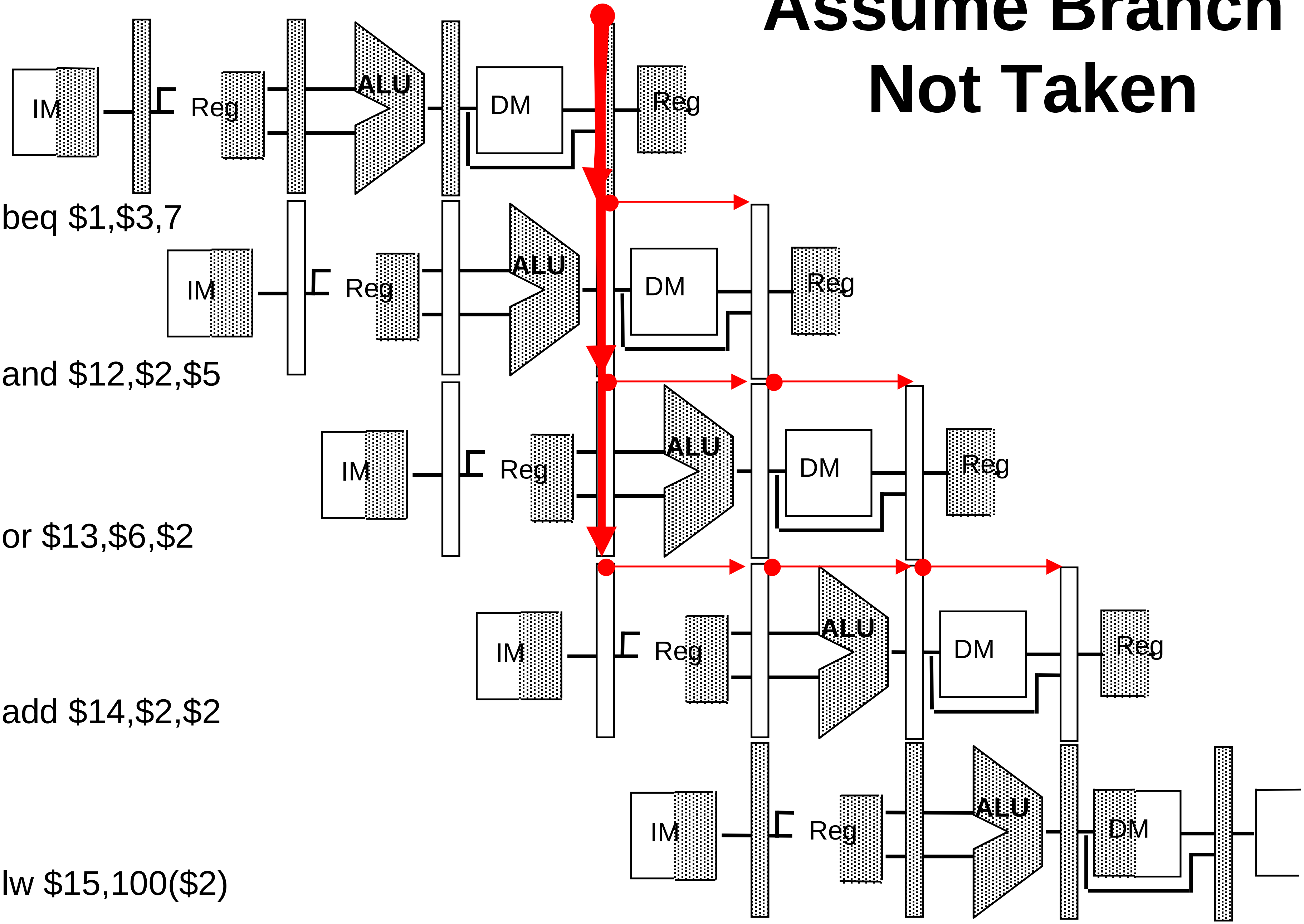
Pipeline Bubbles



Resolving Control Hazards

- Reduce the branch delay
  - Move branch execution hardware to an earlier stage in the pipeline
  - Some MIPS implementation move branches to the ID stage
    - Move the branch adder from MEM stage to ID stage
    - Implement the register comparison in the ID stage, using XOR and AND logic
    - IF.Flush control signal sets instruction bits to a “nop”
  - MIPS R2000/R3000 uses branch delay slot
- Branch Prediction
  - Simple: Assume branch not taken

Assume Branch Not Taken



40 beq $1,$3,7

44 and $12,$2,$5

48 or $13,$6,$2

52 add $14,$2,$2

72 lw $15,100($2)

Assume Branch Not Taken

Implementation

Continue to execute the 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 the control signals, like a stall
EX: Zero the control signals, like a stall

Dynamic Branch Prediction

- One approach
  - Look up branch address to see the previous target
  - Assume next target will be same as previous one
- Implementation
  - Branch prediction buffer
    - table indexed by low-order bits of the branch address
  - 1-bit: marks taken vs. not-taken
  - 2-bit: state machine
    - must be wrong twice before prediction is changed
    - interacts better with loop structures (does guess wrong on the first loop iteration)

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 "not taken" branches, to neutralize the affects of all instructions in the pipe.
>
> Load the exception address into the PC (this may be hard).

Imprecise exceptions.

Pipelining

Pipelining improves performance in certain areas

Improves throughput

Latency mostly unaffected

Pipelining is great, but not perfect. More pipelining

Increases impact of hazards (both data & control)