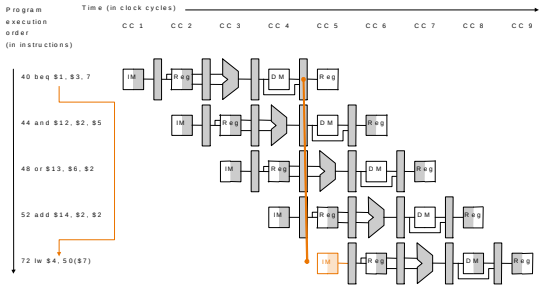
Branch Hazards

Branch Hazards

- In our pipeline decision about whether a branch instruction should branch or not is not made to Mem stage
- Know as a *control hazard*
- Stalling while resolving a control hazard very inefficient
- Can predict:
  - branch not taken
  - outcome of branch dynamically
- Also look at ways to reduce the delay of branches

Branch Hazards



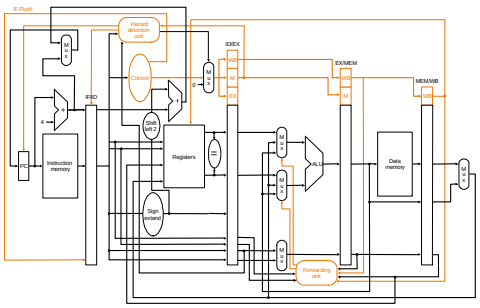
Branch not taken

- Assume branch not taken and continue to fetch instructions until control dependence determined
  - if branch taken then no penalty
  - if branch taken must discard instructions after branch in pipeline
  - if right 50% of time then cost of control hazards halved

Reducing the Delay of Branches

- can reduce cost of mis-predicted branches by moving them earlier in pipeline
  - currently on mem stage
  - Many MIPS implementations move it to decode stage
  - Have to:
    - move the branch address calculation to the ID stage
    - Make the branch decision in the same stage

Reducing Delay of Branches

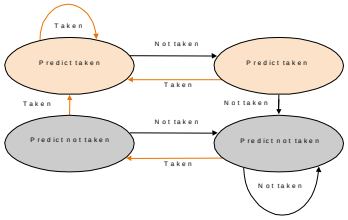


Dynamic Branch Prediction

- Look at branch history to determine whether a branch should be taken
- One approach is to use a branch prediction buffer:
  - a small memory indexed by the LSBs of the address,
  - each location contains a bit that determines whether the branch was taken last time or not
  - This scheme is likely to mis-predict twice in a row when a branch that is almost always taken is not taken

Two-bit prediction schemes

- A prediction must be wrong twice before it is changed



Superscalar and Dynamic Pipelining

- Three main approaches to making processor go even faster:
  - *Super-pipelining* – break pipelines into more stages so CCT can be decreased
    - *e.g. Alpha pipeline with nine stages*
  - *Superscalar* – replicate internal components so that can launch multiple instructions per clock cycle
  - *Dynamic Pipelining* – Hardware schedules instructions so as to avoid having to stall the pipeline

```
lw   $t0, 20($s2)
addu $t1, $t0, $t2
sub  $s4, $s4, $t3
slti $t5, $s4, 20
```

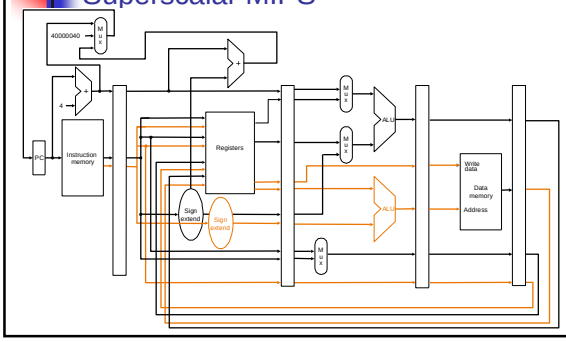
Superscalar MIPS

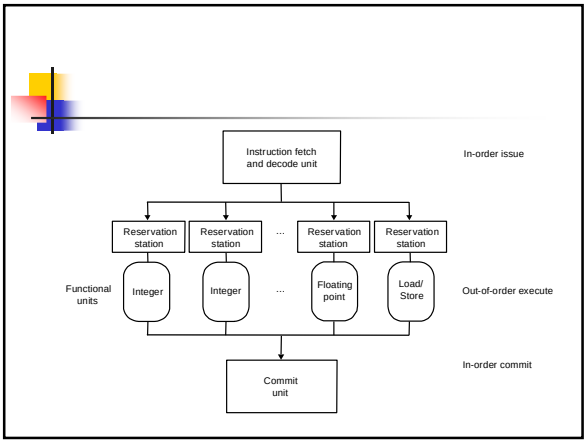
- Assume 2 instructions can be issued per clock
  - one integer or branch
  - one load or store
- Assume instruction pairs are aligned on 64-bit boundary

Superscalar MIPS

instruct type	Pipeline stages							
ALU or Bra	IF	ID	EX	MEM	WB			
mem	IF	ID	EX	MEM	WB			
ALU or Bra		IF	ID	EX	MEM	WB		
mem		IF	ID	EX	MEM	WB		
ALU or Bra			IF	ID	EX	MEM	WB	
mem			IF	ID	EX	MEM	WB	
ALU or Bra				IF	ID	EX	MEM	WB
mem				IF	ID	EX	MEM	WB

Superscalar MIPS




Instruction fetch and decode unit
In-order issue
Reservation station
Reservation station
...
Reservation station
Reservation station
Functional units
Integer
Integer
...
Floating point
Load/ Store
Out-of-order execute
In-order commit
Commit unit