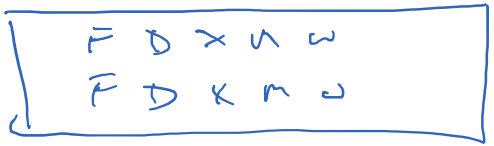
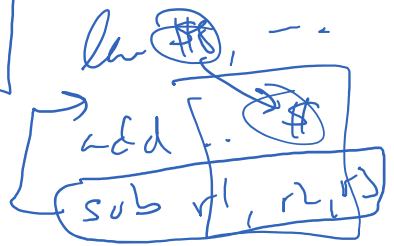
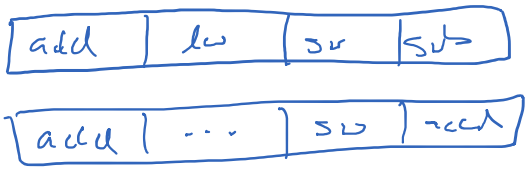
Part II: The Fancy Stuff in Real (Fast) Machines

Pipelining in Today's Most Advanced Processors

- Not fundamentally different than the techniques we discussed
- Deeper pipelines
- Pipelining is combined with
  - superscalar execution
  - out-of-order execution
  - VLIW (very-long-instruction-word)

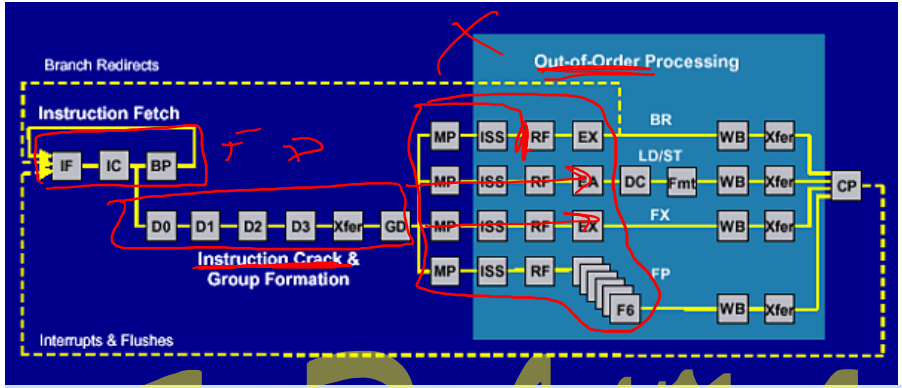




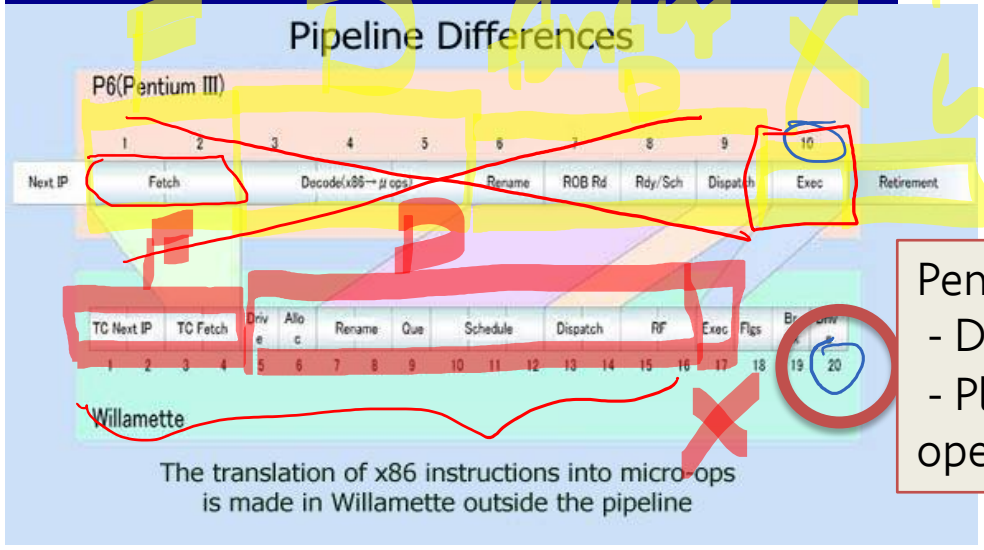


Deeper Pipelines

- Power 4



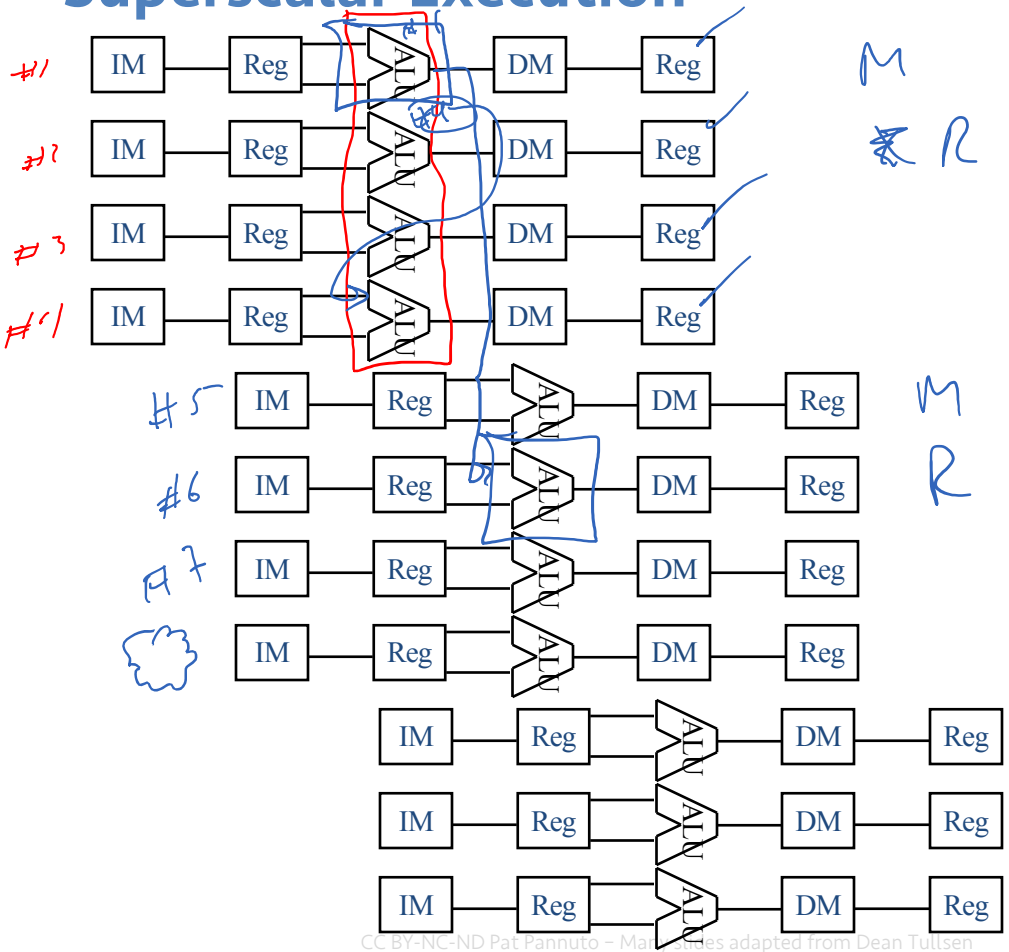
- Pentium 3
- Pentium 4



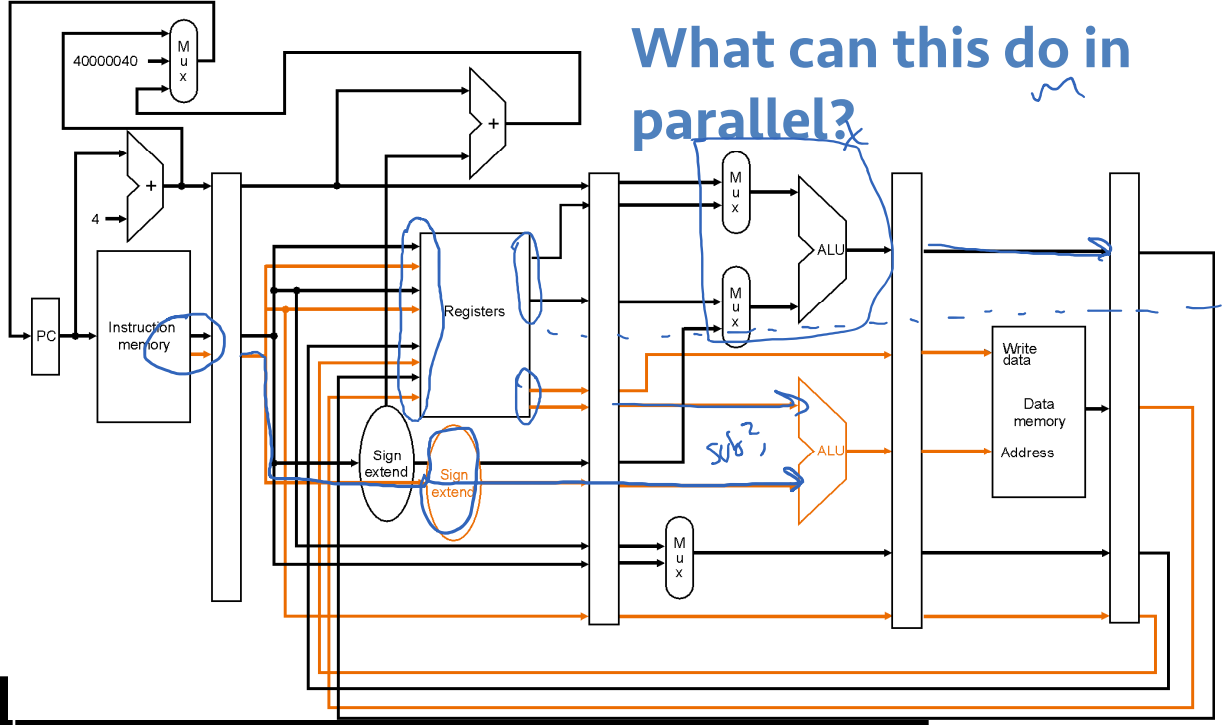
where are branches resolved?

Pentium 4 "Prescott"
- Deeper still: 31 stages!
- Planned for up to 5 GHz operation! (scrapped)

Superscalar Execution

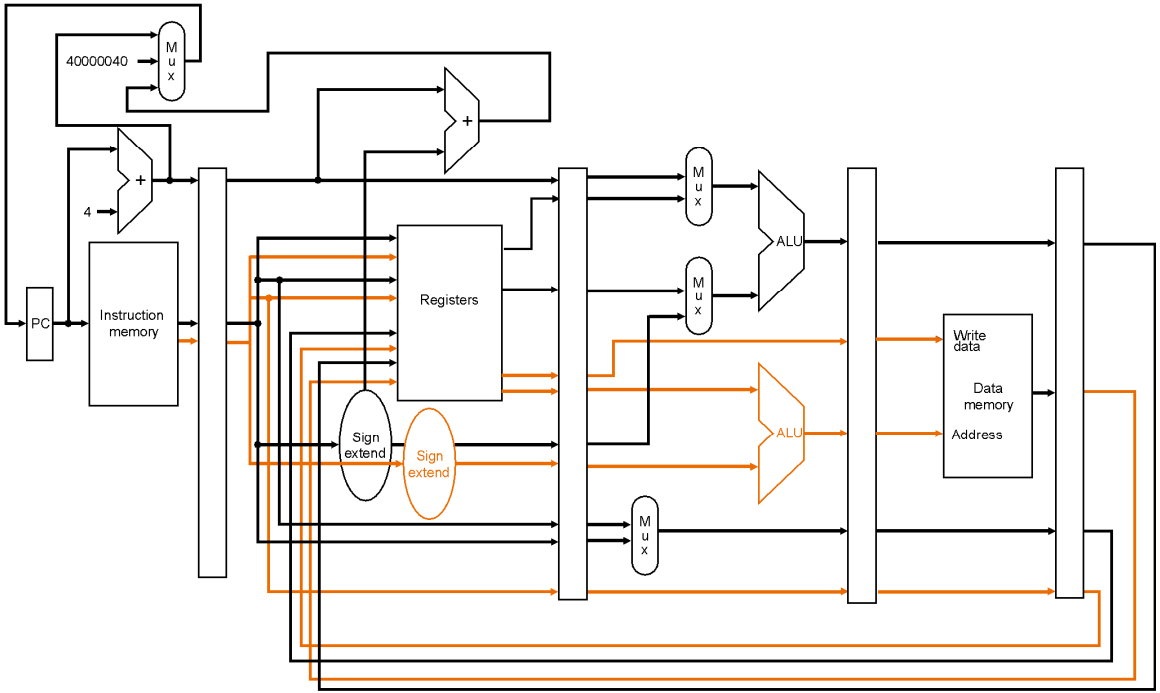


What can this do in parallel?



Selection	
A	Any two instructions
B	Any two *independent* instructions
C	An arithmetic instruction and a memory instruction
D	Any instruction and a memory instruction
E	None of the above

A modest superscalar MIPS



- what can this machine do in parallel?
- what other logic is required?
- Represents earliest superscalar technology (eg, circa early 1990s)

Superscalar Execution

- To execute four instructions in the same cycle, we must find four independent instructions

Superscalar Execution

- To execute four instructions in the same cycle, we must find four independent instructions
- If the four instructions fetched are *guaranteed by the compiler* to be independent, this is a *VLIW* machine

Superscalar Execution

- To execute four instructions in the same cycle, we must find four independent instructions
- If the four instructions fetched are guaranteed by the compiler to be independent, this is a *VLIW* machine
- If the four instructions fetched are only executed together if hardware confirms that they are independent, this is an *in-order superscalar* processor.

Superscalar Execution

- To execute four instructions in the same cycle, we must find four independent instructions
- If the four instructions fetched are guaranteed by the compiler to be independent, this is a *VLIW* machine
- If the four instructions fetched are only executed together if hardware confirms that they are independent, this is an *in-order superscalar* processor.
- If the hardware actively finds four (not necessarily consecutive) instructions that are independent, this is an *out-of-order superscalar* processor.

Superscalar Execution

- To execute four instructions in the same cycle, we must find four independent instructions
- If the four instructions fetched are guaranteed by the compiler to be independent, this is a *VLIW* machine
- If the four instructions fetched are only executed together if hardware confirms that they are independent, this is an *in-order superscalar* processor.
- If the hardware actively finds four (not necessarily consecutive) instructions that are independent, this is an *out-of-order superscalar* processor.
- What do you think are the tradeoffs?

Superscalar Scheduling

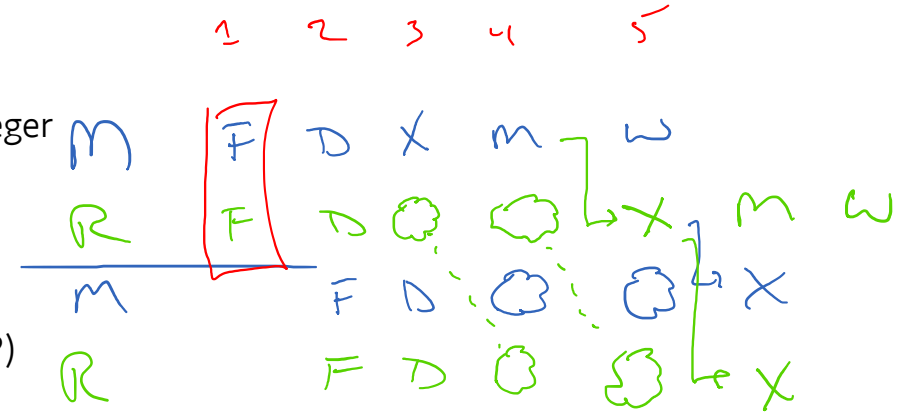
- Assume in-order, 2-issue, ld-store followed by integer

```
lw   $6,   36($2)
add  $5,   $6, $4
lw   $7, 1000($5)
sub  $9, $12, $5
```

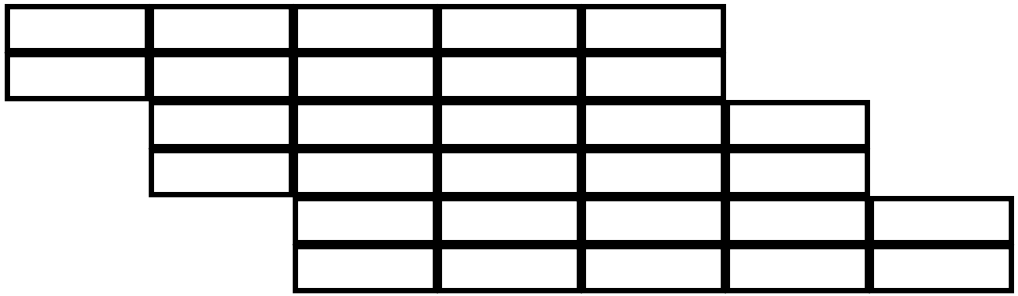
- Assume 4-issue, in-order, any combination (VLIW?)

```
lw   $6, 36($2)
add  $5,  $6, $4
lw   $7, 1000($5)
sub  $9, $12, $5
sw   $5,  200($6)
add  $3,  $9, $9
and $11,  $7, $6
```

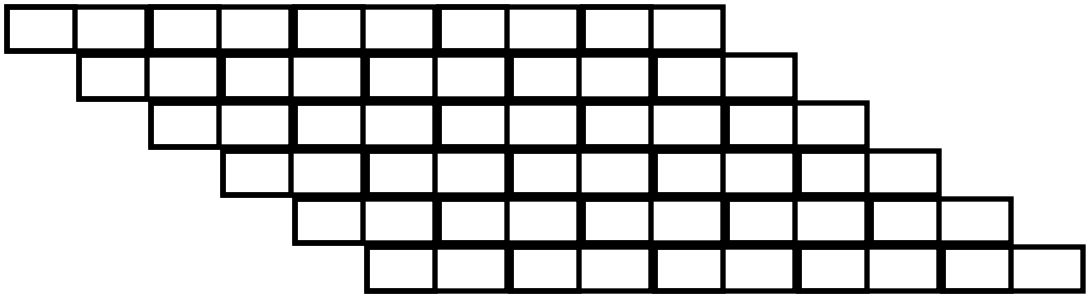
- When does each instruction begin execution?



Superscalar vs. superpipelined



(multiple instructions in the same stage, same clock rate as scalar)



(more total stages, faster clock rate)

Dynamic Scheduling
aka, Out-of-Order Scheduling

- Issues (begins execution of) an instruction as soon as all of its dependences are satisfied, even if prior instructions are stalled. (assume 2-issue, any combination)

```
lw   $6,   36($2)
add  $5,  $6, $4
lw   $7, 1000($5)
sub  $9, $12, $8
sw   $5,  200($6)
add  $3,  $9, $9
and $11,  $5, $6
```

Reservation Stations

(other pieces: ROB, RAT, RRAT.. CSE 148 covers these!)

- Are a mechanism to allow dynamic scheduling (out of order execution)