High Performance Processors
CS 740
Sept. 29, 1997

Multicycle Instructions

- Integer multiplication / division
- Floating Point

Overview of Processor Classes

- Intel Microprocessor Progression

Multicycle instructions

MIPS R3000 Execution Times

- Measured in clock cycles

Operation	Integer	Single	Double
add / sub	1	2	2
multiply	~12	4	5
divide	~35	12	19

H&P Dynamic Instruction Counts

Operation	*Int. Benchmark*	*FP Benchmark*	
		Integer	*FP*
add / sub	14%	11%	14%
multiply	< 0.1%	< 0.1%	13%
divide	< 0.1%	< 0.1%	1%

MIPS Integer Hardware (Guess)

Extra Logic for EX Stage

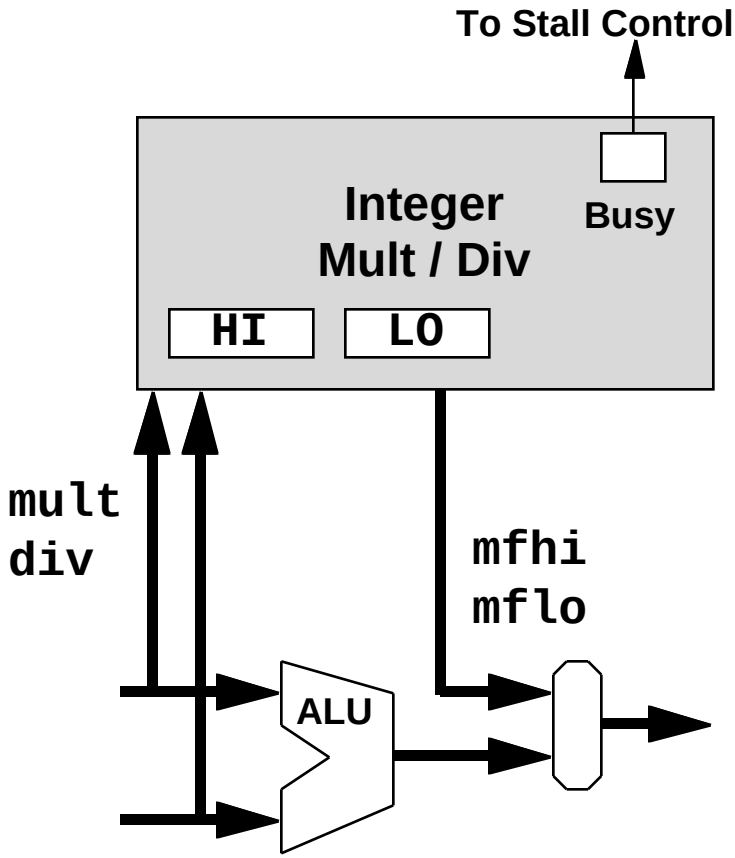
- **HI, LO registers**
- **Sequential multiplier / divider**
  - 3 bits / iteration for multiply
  - 1 bit / iteration for division

Mult / Div Instruction

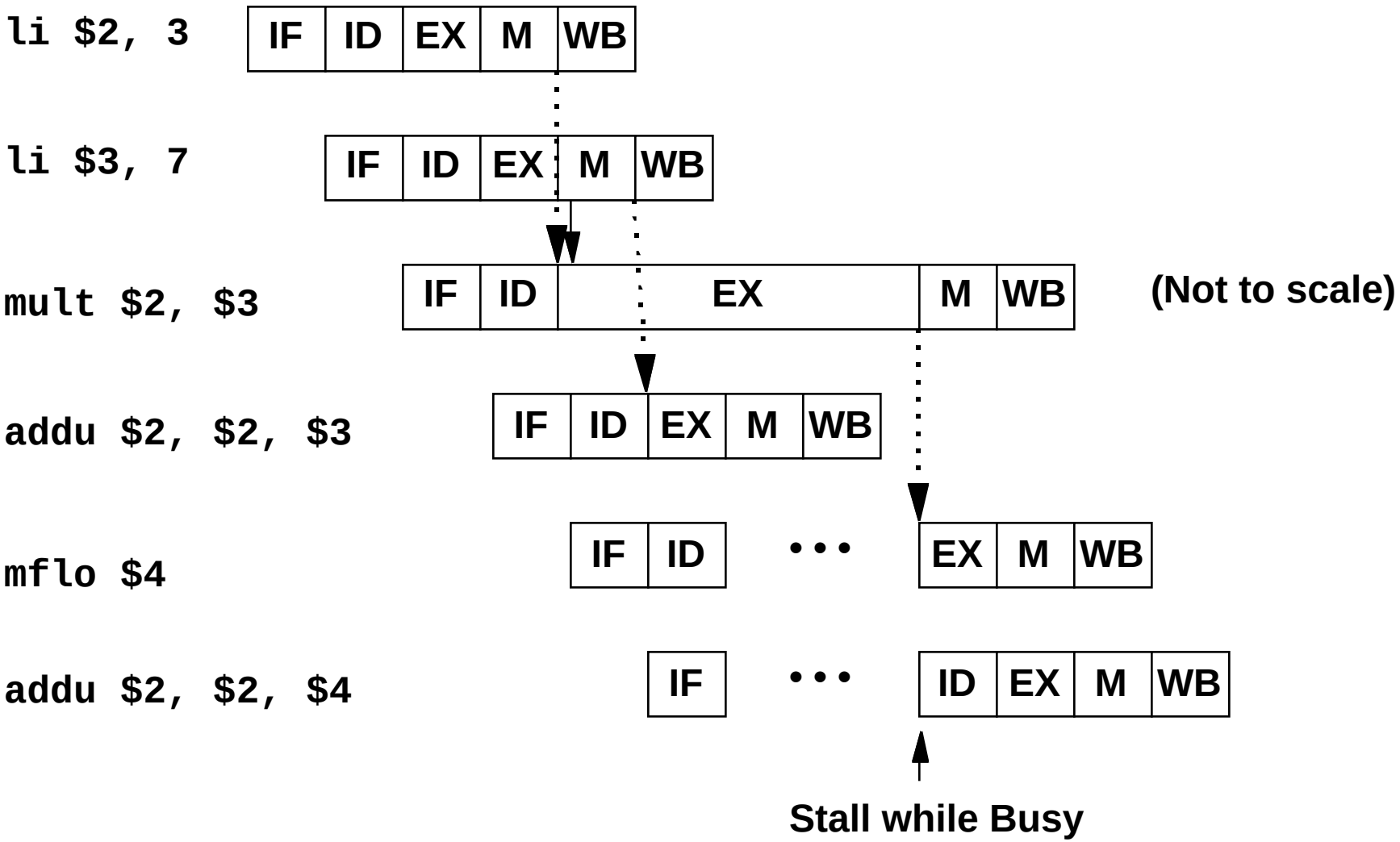
- **Stall while Busy**
  - Only 1 mult / div operation at a time
- **Load operands in unit**
  - Usual bypass paths

Mfhi / Mflo Instruction

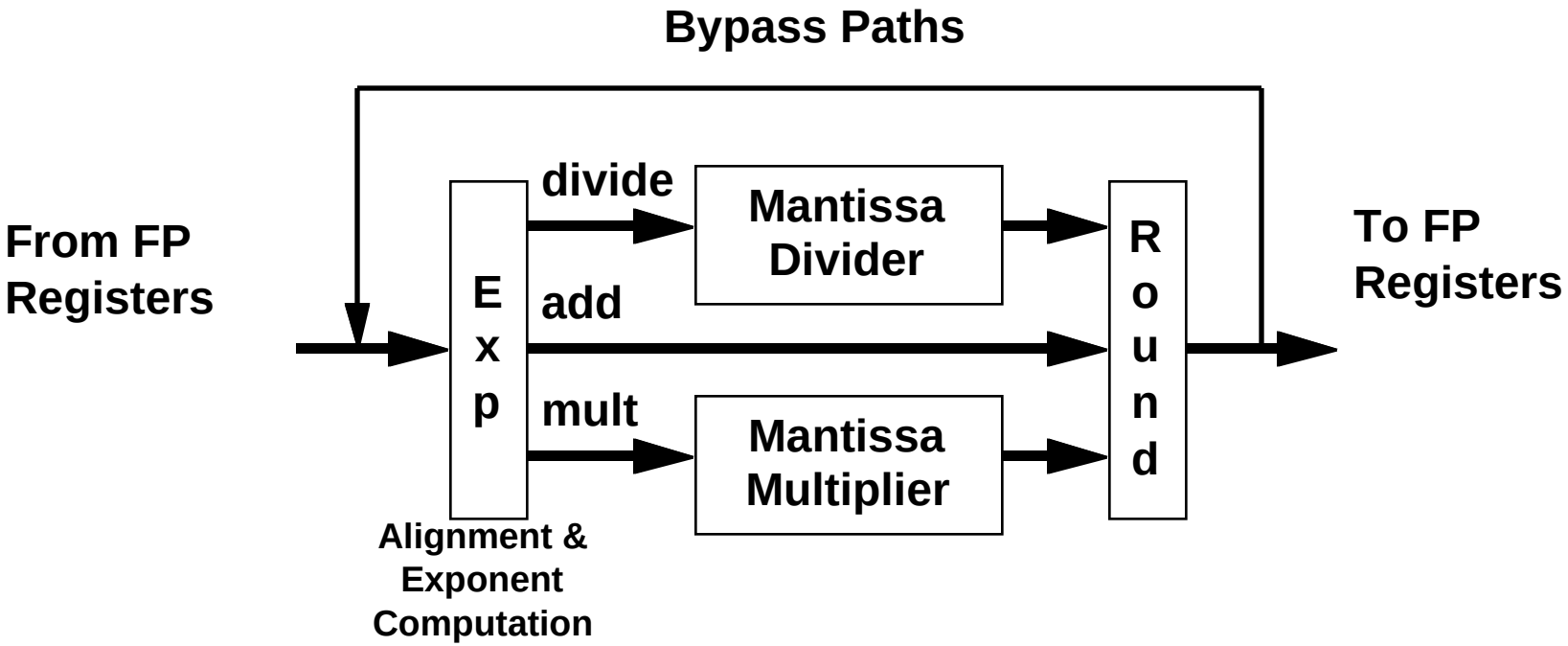
- **Stall while Busy**
- **Register becomes EX output**



Multiply Timing Example



MIPS Floating Point Hardware



Independent Hardware Units

- Can concurrently execute add, divide, multiply
- Except that all share exponent and rounding units
- Independent of integer operations

Control Logic

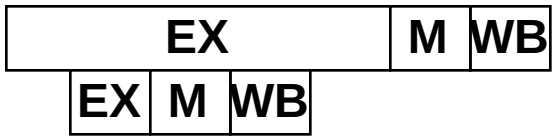
Busy Flags

- **One per hardware unit**
- **One per FP register**
  - Destination of currently executing operation

Stall Instruction in ID if:

- **Needs unit that is not available**
- **Source register busy**
  - Avoids RAW (Read-After-Write) hazard
- **Destination register busy**
  - Avoids WAW hazard

```
mul.d $f4, $f0, $f2
add.d $f4, $f0, $f2
```



Bypass paths

- **Similar to those in integer pipeline**

FP Timing Example

```
add.d $f0, $f2, $f4
mul.d $f6, $f0, $f2
```

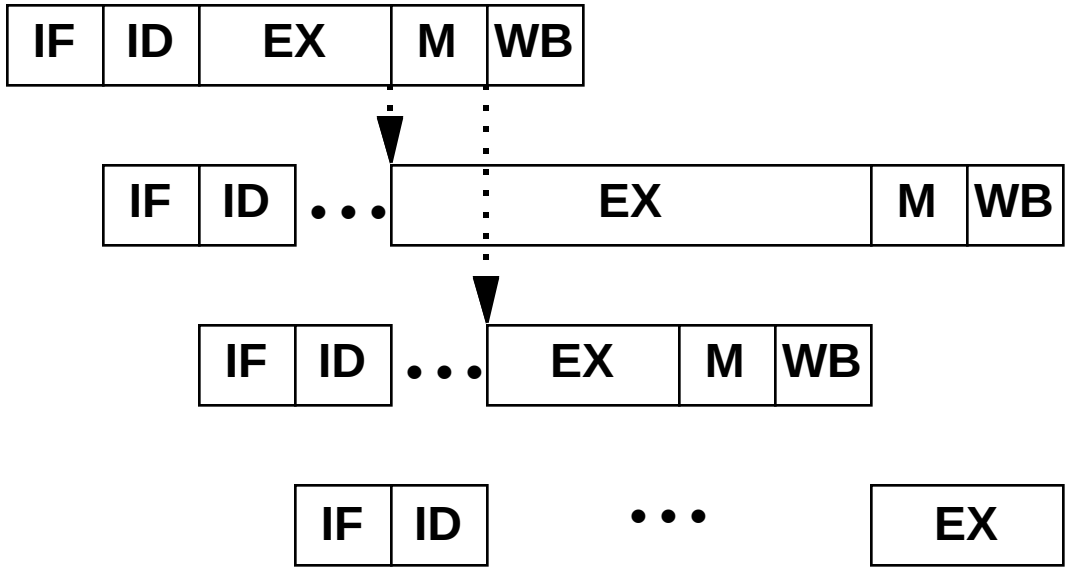
RAW Stall

```
add.d $f8, $f0, $f2
```

Structural Stall

```
add.d $f6, $f0, $f2
```

WAW Stall



MIPS Wrap-Up

MIPS Pipeline Characteristics

- **In-order issue**
  - Instructions fetched and decoded in program order
- **Out-of-order completion**
  - Slow instructions may complete after ones that are later in program order

Performance Opportunities

- **Compiler optimizations to expose potential parallelism**
- **Schedule code to use multiple functional units**
  - Must understand idiosyncracies of pipeline structure

Intel x86 Processors

Processor	Year	Transistors	MHz	Spec92 (Int/FP)	Spec95 (Int/FP)
8086	‘78	29K	4		
Basis of IBM PC & PC-XT					
i286	‘83	134K	8		
Basis of IBM PC-AT					
i386	‘86	275K	16		
	‘88		33	6 / 3	
i486	‘89	1.2M	20		
			50	28 / 13	
Pentium	‘93	3.1M	66	78 / 64	
			150	181 / 125	4.3 / 3.0
PentiumPro	‘95	5.5M	150	245 / 220	6.1 / 4.8
			200	320 / 283	8.2 / 6.0
Pentium II	‘97	7.5M	300		11.6 / 6.8
P7 (Merced)	‘98?	14M	?	?	?

Other Processors

Processor	Year	Transistors	MHz	Spec92	Spec95
MIPS R3000	‘88		25	16.1 / 21.7	
(DecStation 5000/120)					
MIPS R5000		3.6M	180		4.1 / 4.4
(Wean Hall SGIs)					
MIPS R10000	‘95	5.9M	200	300 / 600	10.7 / 17.4
(Most Advanced MIPS)					
Alpha 21164a	‘96	9.3M	417	500 / 750	11 / 17
			600		18.4 / 21.3
(Fastest Available)					
Alpha 21264	‘97	15M	500		30 / 60
(Fastest Announced)					

Architectural Performance

Metric

- SpecX92/Mhz: Normalizes with respect to clock speed
- But … one measure of good arch. is how fast can run clock

Sampling

Processor	MHz	SpecInt92	IntAP	SpecFP92	FltAP
i386/387	33	6	0.2	3	0.1
i486DX	50	28	0.6	13	0.3
Pentium	150	181	1.2	125	0.8
PentiumPro	200	320	1.6	283	1.4
MIPS R3000A	25	16.1	0.6	21.7	0.9
MIPS R10000	200	300	1.5	600	3.0
Alpha 21164a	417	500	1.2	750	1.8

x86 ISA Characteristics

Multiple Data Sizes and Addressing Methods

- Recent generations optimized for 32-bit mode

Limited Number of Registers

- Stack-oriented procedure call and FP instructions
- Programs reference memory heavily (41%)

Variable Length Instructions

- First few bytes describe operation and operands
- Remaining ones give immediate data & address displacements
- Average is 2.5 bytes

i486 Pipeline

Fetch

- Load 16-bytes of instruction into prefetch buffer

Decode1

- Determine instruction length, instruction type

Decode2

- Compute memory address
- Generate immediate operands

Execute

- Register Read
- ALU operation
- Memory read/write

Write-Back

- Update register file

486 Pipeline Stage Details

Fetch

- **Moves 16 bytes of instruction stream into code queue**
- **Not required every time**
  - About 5 instructions fetched at once
  - Only useful if don't branch
- **Avoids need for separate instruction cache**

D1

- **Determine total instruction length**
  - Signals code queue aligner where next instruction begins
- **May require two cycles**
  - When multiple operands must be decoded
  - About 6% of "typical" DOS program

486 Stage Details (Cont.)

D2

- **Extract memory displacements and immediate operands**
- **Compute memory addresses**
  - Add base register, and possibly scaled index register
- **May require two cycles**
  - If index register involved, or both address & immediate operand
  - Approx. 5% of executed instructions

EX

- **Read register operands**
- **Compute ALU function**
- **Read or write memory (data cache)**

WB

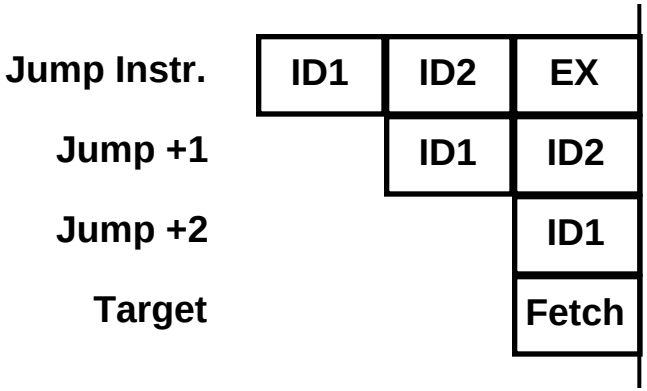
- **Update register result**

486 Data Hazards

Data Hazards

Generated	Used	Handling
ALU	ALU	EX–EX Forwarding
Load	ALU	EX–EX Forwarding
ALU	Store	EX–EX Forwarding
ALU	Eff. Address	(Stall) + EX–ID2 Forwarding

486 Control Hazards



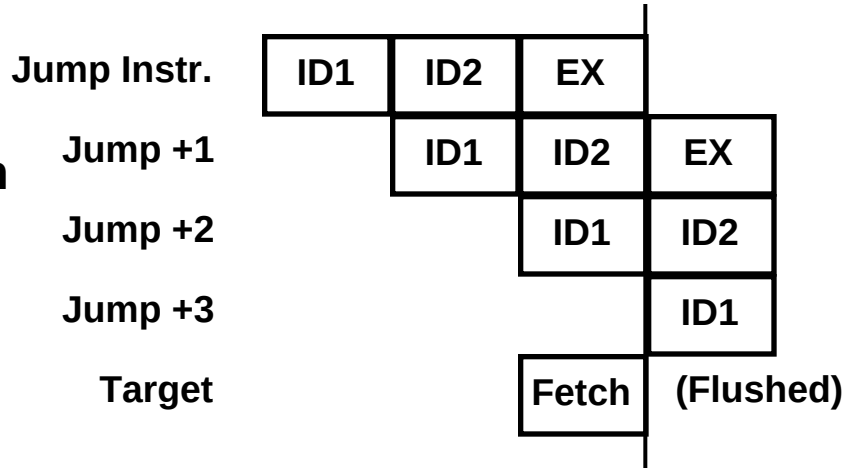
Jump Instruction Processsing

- Continue pipeline assuming branch not taken
- Resolve branch condition in EX stage
- Also speculatively fetch at target during EX stage

486 Control Hazards (Cont.)

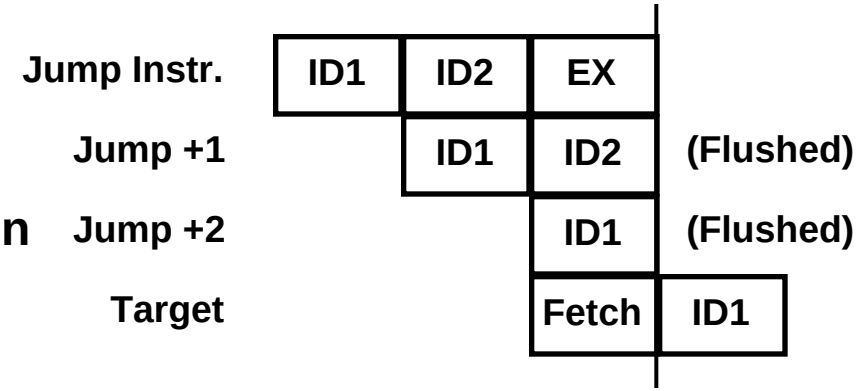
Branch Not Taken

- Allow pipeline to continue.
- Total of 1 cycle for instruction



Branch taken

- Flush instructions in pipe
- Begin ID1 at target.
- Total of 3 cycles for instruction



Comparison to MIPS

Two Decoding Stages

- **Harder to decode CISC instructions**
- **Effective address calculation in D2**

Multicycle Decoding Stages

- **For more difficult decodings**
- **Stalls incoming instructions**

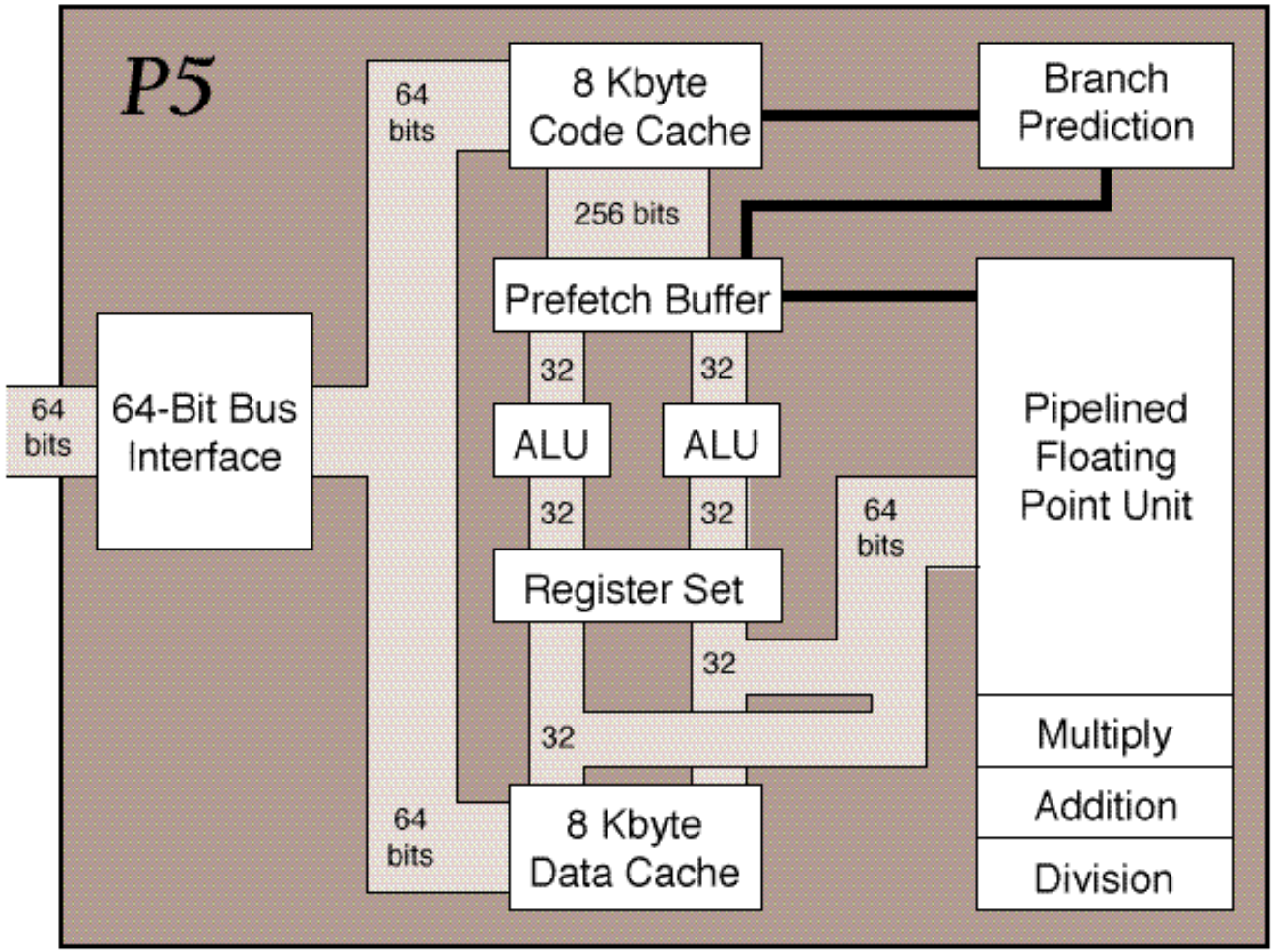
Combined Mem/EX Stage

- **Avoids load stall without load delay slot**
  - But introduces stall for address computation

Poorer Branch Performance

- **Can't make use of a delay slot**
- **Asymmetric performance**

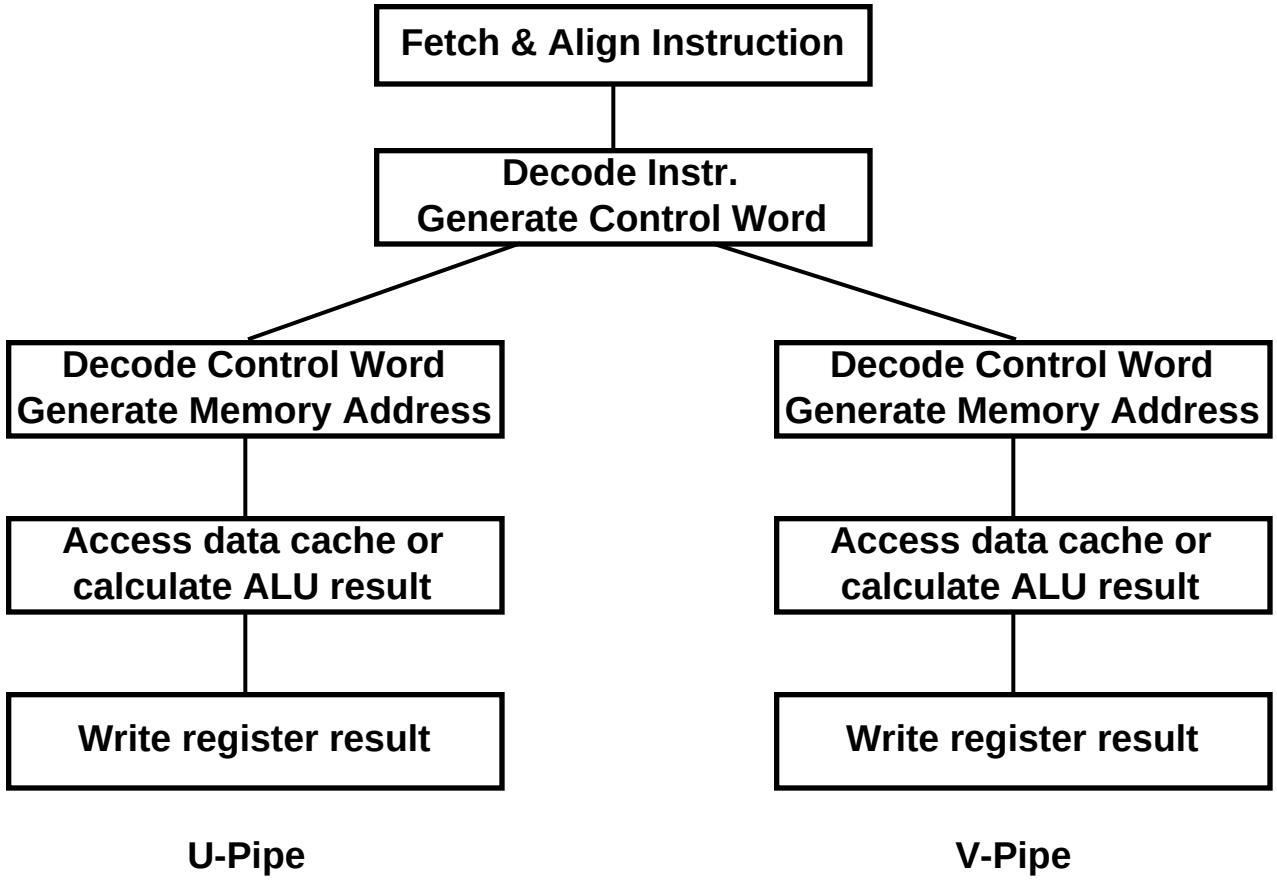
Pentium Block Diagram



Memory Data Bus

(Microcprocessor Report 10/28/92)

Pentium Pipeline



Superscalar Execution

Can Execute Instructions I1 & I2 in Parallel if:

- **Both are "simple" instructions**
  - Don't require microcode sequencing
  - Some operations require U-pipe resources
  - 90% of SpecInt instructions
- **I1 is not a jump**
- **Destination of I1 not source of I2**
  - But can handle I1 setting CC and I2 being cond. jump
- **Destination of I1 not destination of I2**

If Conditions Don't Hold

- **Issue I1 to U Pipe**
- **I2 issued on next cycle**
  - Possibly paired with following instruction

Branch Prediction

Branch Target Buffer

- **Stores information about previously executed branches**
  - Indexed by instruction address
  - Specifies branch destination + whether or not taken
- **256 entries**

Branch Processing

- **Look for instruction in BTB**
- **If found, start fetching at destination**
- **Branch condition resolved early in WB**
  - If prediction correct, no branch penalty
  - If prediction incorrect, lose ~3 cycles
    - » Which corresponds to > 3 instructions
- **Update BTB**

Superscalar Terminology

Basic

Superscalar	**Able to issue > 1 instruction / cycle**
Superpipelined	**Deep, but not superscalar pipeline.**
	E.g., MIPS R5000 has 8 stages
Branch predication	**Logic to guess whether or not branch will be taken, and possibly branch target**

Advanced

Out-of-order	**Able to issue instructions out of program order**
Speculation	**Execute instructions beyond branch points, possibly nullifying later**
Register renaming	**Able to dynamically assign physical registers to instructions**
Retire unit	**Logic to keep track of instructions as they complete.**

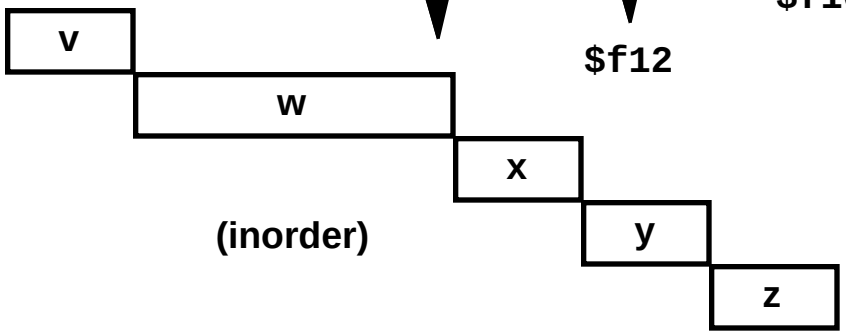
Superscalar Execution Example

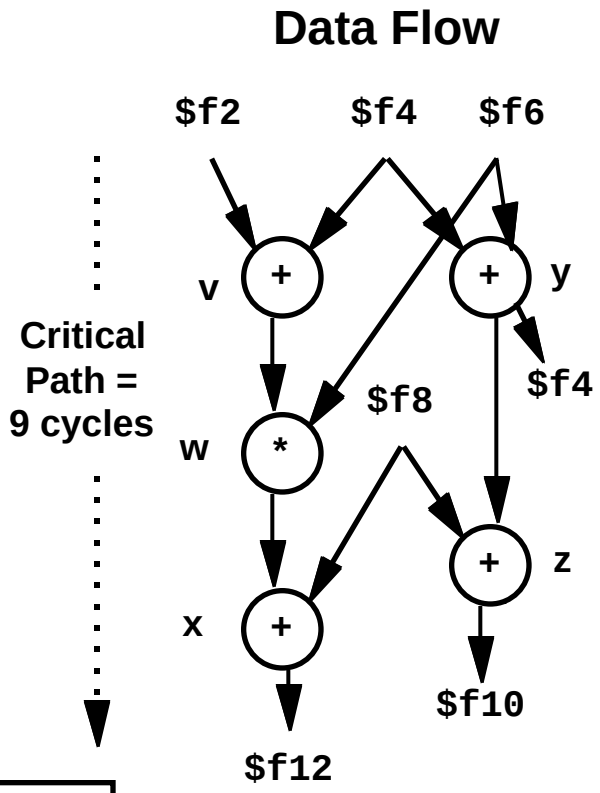
Assumptions

- Single FP adder takes 2 cycles
- Single FP multipler takes 5 cycles
- Can issue add & multiply together
- Must issue in-order

(Single adder, data dependence)
(In order)

```
v:   add.d $f10, $f2,  $f4
w:   mul.d $f10, $f10,  $f6
x:   add.d $f12, $f10, $f8
y:   add.d $f4,  $f4,  $f6
z:   add.d $f10, $f4,  $f8
```



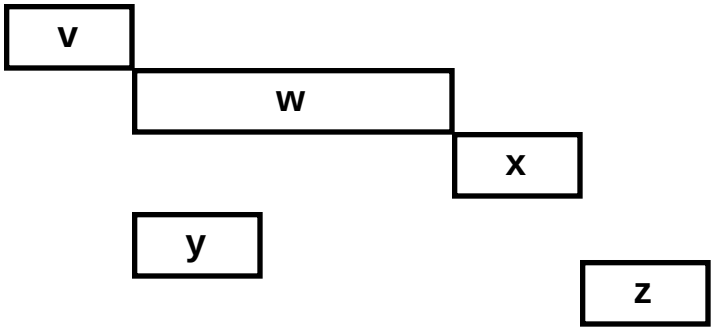


Adding Advanced Features

Out Of Order Issue

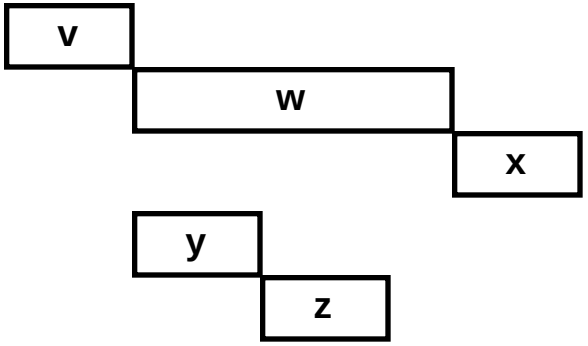
- Can start y as soon as adder available
- Must hold back z until `$f10` not busy & adder available (WAR Hazard)

```
v:   add.d $f10, $f2,  $f4
w:   mul.d $f10, $f10,  $f6
x:   add.d $f12, $f10, $f8
y:   add.d $f4,  $f4,  $f6
z:   add.d $f10, $f4,  $f8
```



With Register Renaming

```
v:   add.d $f10a, $f2,  $f4
w:   mul.d $f10a, $f10a,  $f6
x:   add.d $f12, $f10a, $f8
y:   add.d $f4,  $f4,  $f6
z:   add.d $f10, $f4,  $f8
```



Pentium Pro (P6)

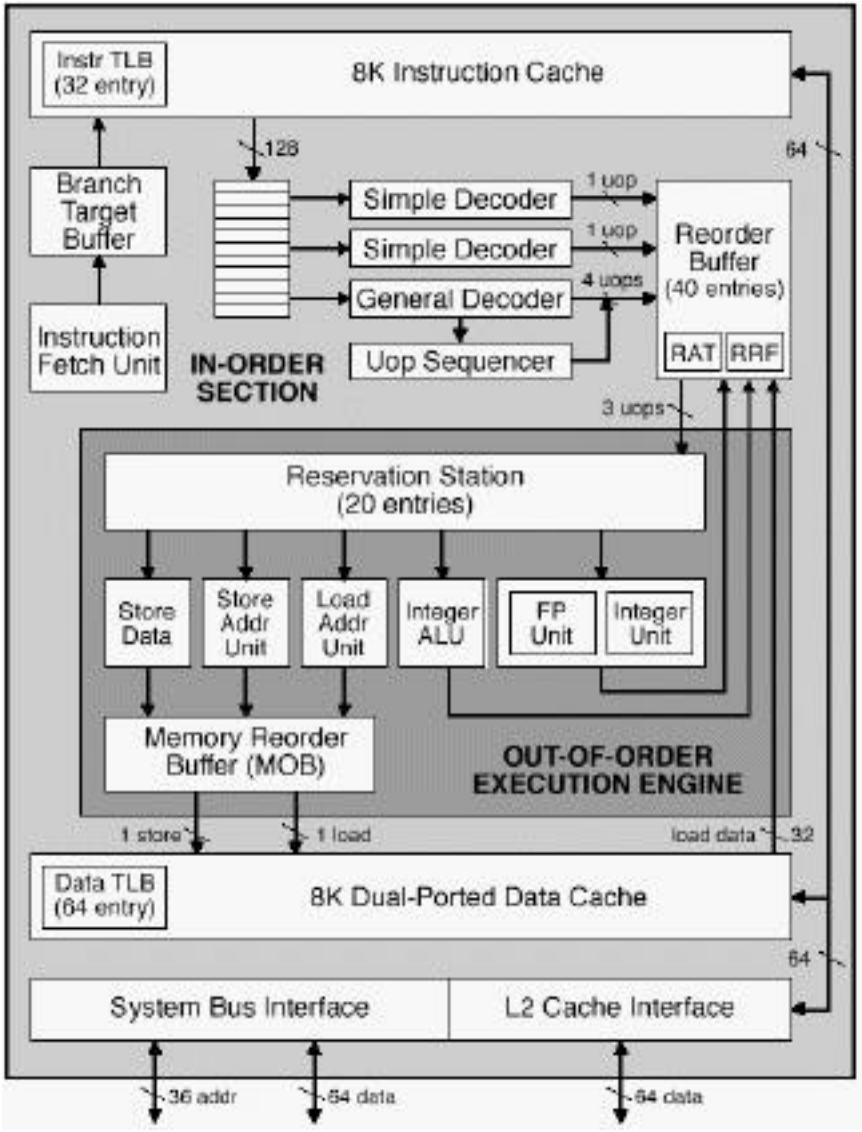
History

- **Announced in Feb. ‘95**
- **Delivering in high end machines now**

Features

- **Dynamically translates instructions to more regular format**
  - Very wide RISC instructions
- **Executes operations in parallel**
  - Up to 5 at once
- **Very deep pipeline**
  - 12–18 cycle latency

PentiumPro Block Diagram



Microprocessor Report
2/16/95

PentiumPro Operation

Translates instructions dynamically into “Uops”

- **118 bits wide**
- **Holds operation, two sources, and destination**

Executes Uops with “Out of Order” engine

- **Uop executed when**
  - Operands available
  - Functional unit available
- **Execution controlled by “Reservation Stations”**
  - Keeps track of data dependencies between uops
  - Allocates resources

Branch Prediction

Critical to Performance

- **11–15 cycle penalty for misprediction**

Branch Target Buffer

- **512 entries**
- **4 bits of history**
- **Adaptive algorithm**
  - Can recognize repeated patterns, e.g., alternating taken–not taken

Handling BTB misses

- **Detect in cycle 6**
- **Predict taken for negative offset, not taken for positive**
  - Loops vs. conditionals

Processor Comparisons

PROCESSORS FOR WORKSTATIONS AND SERVERS											
	Max Clock Speed	Cache Size	Supply Voltage	Max Power	Transistor Count	IC Process	Die Size	Est Mfg Cost*	SPEC95b int/fp	List Price	Avail-ability
Digital 21164	500 MHz	8K/8K/96K	2.0 V	25 W	9.3 million	0.35µ 4M	209 mm²	$150	12.6/18.3	$1,450	now
Digital 21264	>500 MHz	64K/64K	2.0 V	60 W	15 million	0.35µ 6M	300 mm²	$300	30/60	N.D.	4Q97
Fuj. TurboSparc	170 MHz	16K/16K	3.3 V	9 W	3.0 million	0.35µ 4M	132 mm²	$50	3.5/3.0	$499	now
HP PA-7300LC	160 MHz	64K/64K	3.3 V	15 W	9.2 million	0.5µ 4M	259 mm²	$95	5.5/7.3	not sold	now
HP PA-8000	180 MHz	none	3.3 V	>40 W	3.9 million	0.5µ 4M	345 mm²	$290	10.8/18.3	not sold	now
IBM P2SC	135 MHz	32K/128K	2.5 V	30 W	15 million	0.29µ 4M	335 mm²	$375	5.5/14.5	not sold	now
MIPS R5000	180 MHz	32K/32K	3.3 V	10 W	3.6 million	0.35µ 3M	84 mm²	$25	4.0/3.7	$365	now
MIPS R7000	300 MHz	288K(1)	3.3 V	13 W	N.D.	0.25µ 4M	80 mm²	$35	10/10	N.D.	2H97
MIPS R10000	200 MHz	32K/32K	3.3 V	30 W	5.9 million	0.35µ 4M	298 mm²	$160	8.9/17.2	$3,000	now
Sun UltraSparc-2	250 MHz	16K/16K	2.5 V	20 W	3.8 million	0.29µ 5M	149 mm²	$90	8.5/15	$1,995	limited
PROCESSORS FOR PCS AND WORKSTATIONS											
	Max Clock Speed	Cache Size	Supply Voltage	Max Power	Transistor Count	IC Process	Die Size	Est Mfg Cost*	SPEC95b int/fp	List Price	Avail-ability
Exponential x704	533 MHz	2K/2K/32K	3.6 V	85 W	2.7 million	0.5µ 5M(2)	150 mm²	$90	12/10	$1,000*	2Q97
PowerPC 603e	240 MHz	16K/16K	2.5 V	6 W	2.6 million	0.35µ 4M	79 mm²	$30	5.5/4*	$408	now
PowerPC 604e	225 MHz	32K/32K	2.5 V	24 W	5.1 million	0.35µ 4M	148 mm²	$60	8/7*	$533	now
Intel Pentium	200 MHz	8K/8K	3.3 V	17 W	3.3 million	0.35µ 4M(3)	90 mm²	$40	5.5/2.9	$509	now
Intel P55C	200 MHz	16K/16K	2.8 V	16 W	4.5 million	0.28µ 4M	140 mm²	$50	6/3*	N.D.	1Q97
Intel PPro	200 MHz	8K/8K	3.3 V	35 W†	5.5 million	0.35µ 4M(3)	196 mm²	$145†	8.2/6.0†	$525†	now
Intel Klamath	266 MHz*	16K/16K	2.8 V*	N.D.	7.5 million	0.28µ 4M	203 mm²	$80	11/7*	N.D.	2Q97*

Microprocessor Report 12/30/96

Challenges for Processor Design

Diminishing Returns on Cost vs. Performance

- **Superscalar processors require instruction level parallelism**
- **Many programs limited by sequential dependencies**

Getting Design Correct Difficult

- **Verfication team larger than design team**
- **Devise tests for interactions between concurrent instructions**
  - May be 80 executing at once

Instruction Sets Get in the Way (especially x86)

- **Not enough registers**
- **Too many memory references**
- **(Apparently) Intel is going to new instruction set for Merced**
  - IA-64, joint with HP
  - Will dynamically translate existing x86 binaries