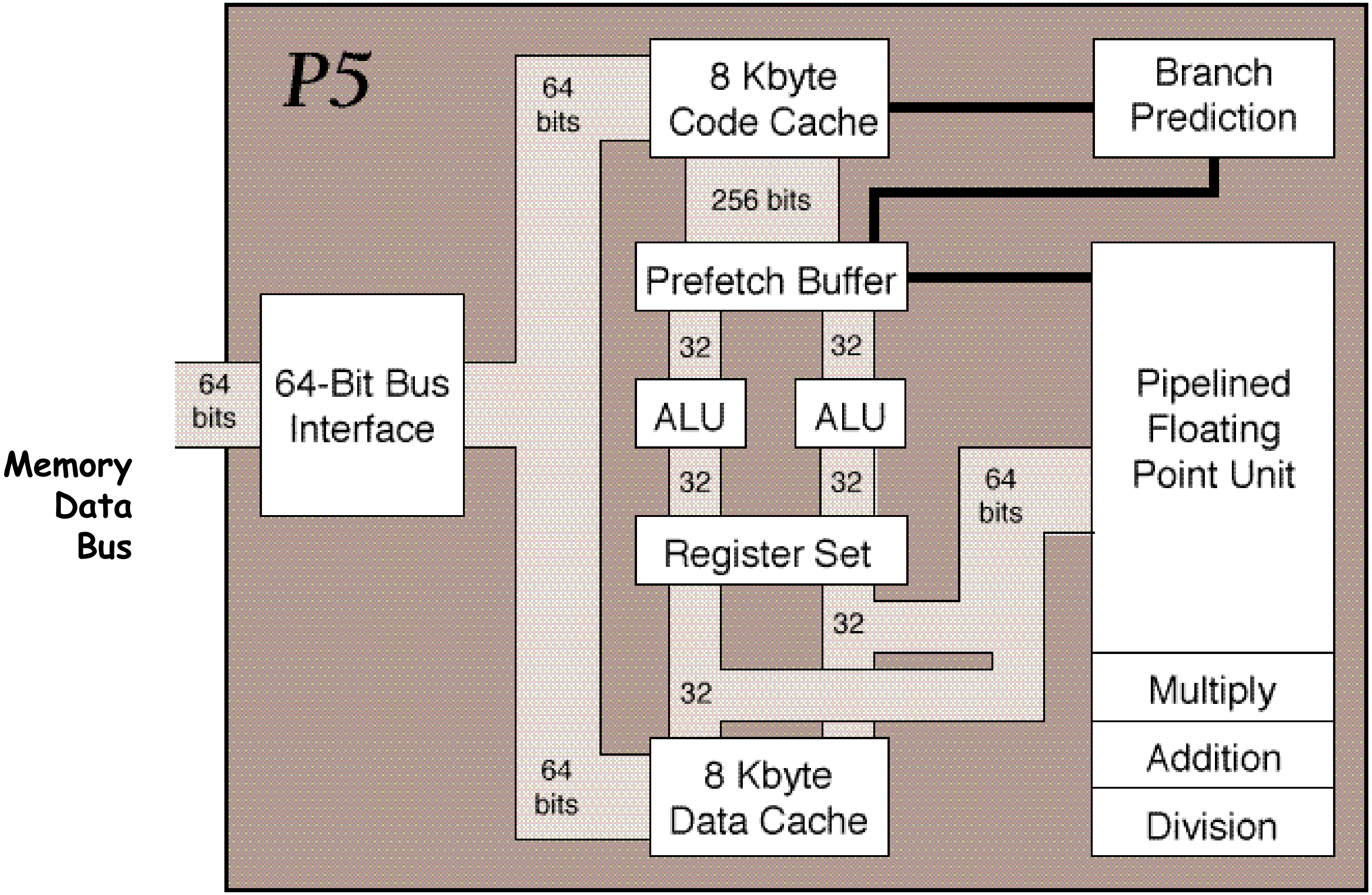
ELEC3020: Lecture 4-2

The Pentium and Pentium Pro Full Superscalar Execution

With thanks to CS370, Superscalar Processing at CMU

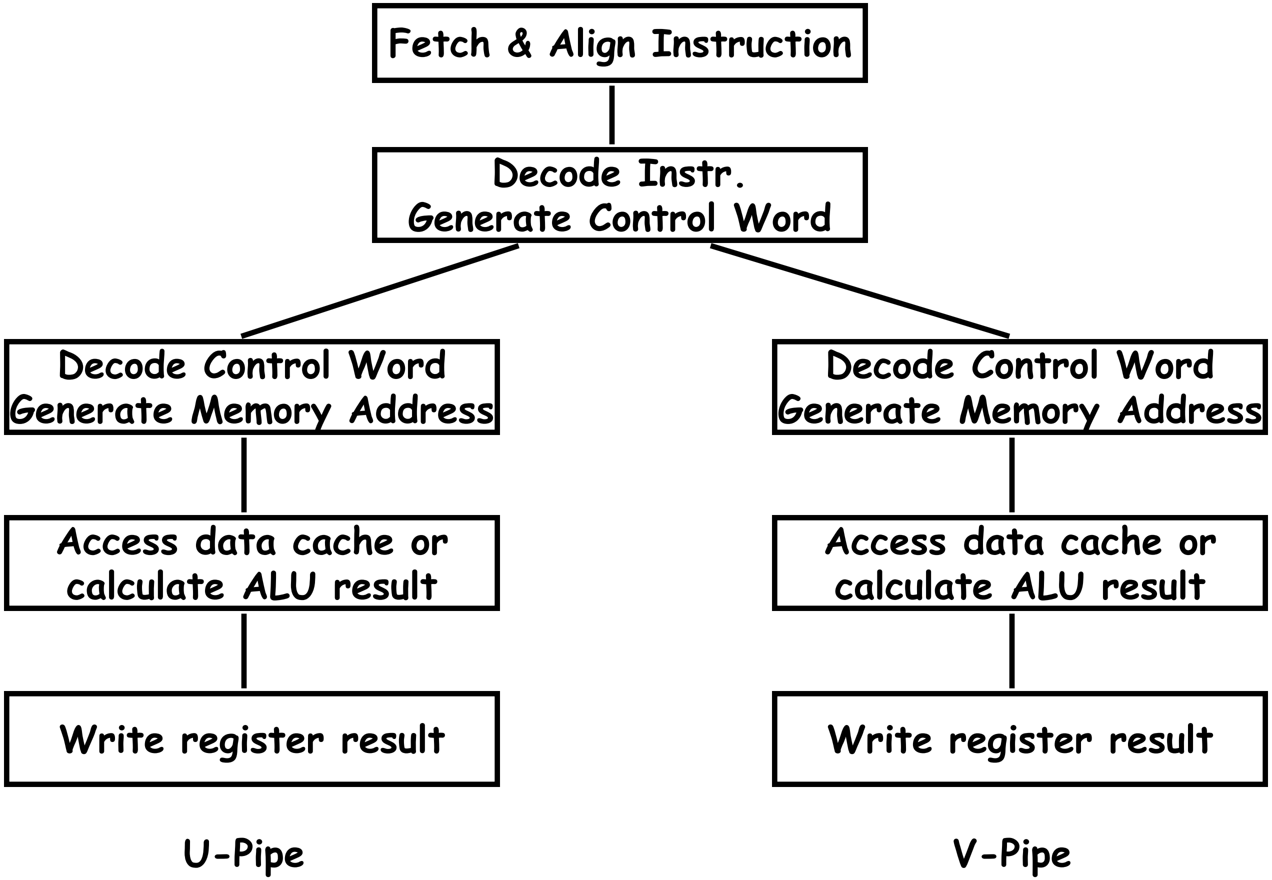
Pentium Block Diagram



(Microcprocessor Report 10/28/92)

With thanks to CS370, *Superscalar Processing* at CMU

Pentium Pipeline



Fetch & Align Instruction

Decode Instr.
Generate Control Word

Decode Control Word
Generate Memory Address

Access data cache or
calculate ALU result

Write register result

U-Pipe

Decode Control Word
Generate Memory Address

Access data cache or
calculate ALU result

Write register result

V-Pipe

With thanks to CS370, *Superscalar Processing* at CMU

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

With thanks to CS370, *Superscalar Processing* at CMU

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

With thanks to CS370, *Superscalar Processing* at CMU

Superscalar Terminology

- Basic

Superscalar	Able to issue > 1 instruction / cycle
Superpipelined	Deep, but not superscalar pipeline. E.g., MIPS R5000 has 8 stages
Branch prediction	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.

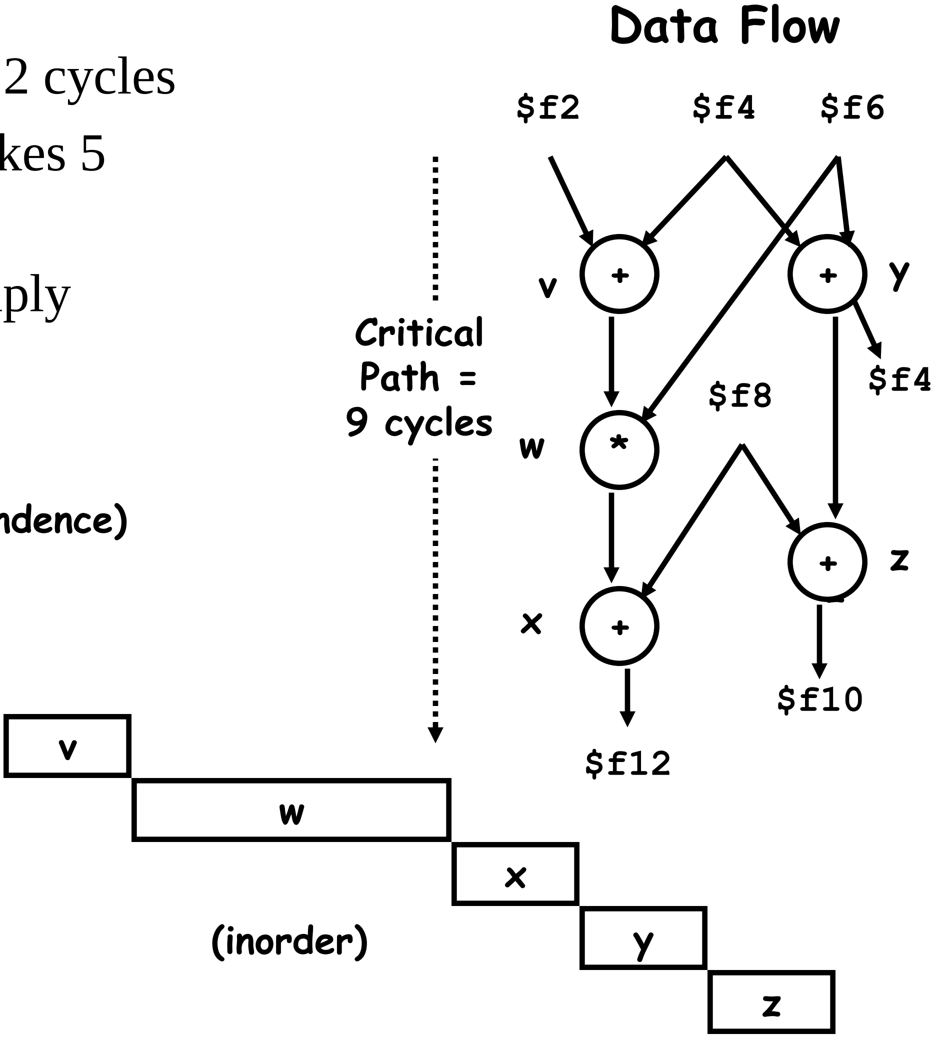
Superscalar Execution Example

- 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:   addt  $f2, $f4, $f10
w:   mult $f10, $f6, $f10
x:   addt $f10, $f8, $f12
y:   addt  $f4, $f6,  $f4
z:   addt  $f4, $f8, $f10
```

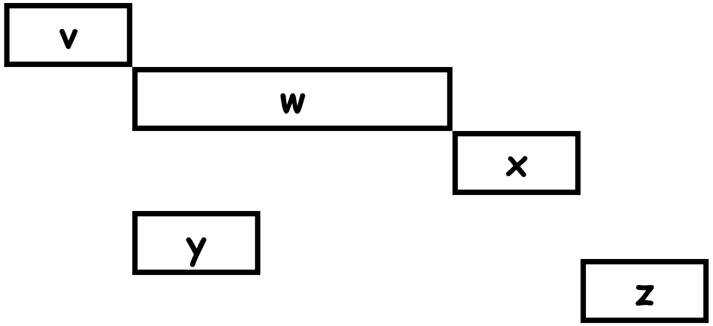


With thanks to CS370, *Superscalar Processing* at CMU

Adding Advanced Features

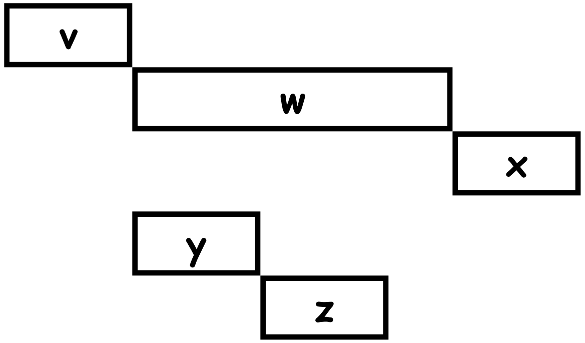
- Out Of Order Issue
  - Can start y as soon as adder available
  - Must hold back z until `$f10` not busy & adder available

```
v:    addt  $f2, $f4, $f10
w:    mult $f10, $f6, $f10
x:    addt $f10, $f8, $f12
y:    addt  $f4, $f6,  $f4
z:    addt  $f4, $f8, $f10
```



- With Register Renaming

```
v:    addt  $f2, $f4, $f10a
w:    mult $f10a, $f6, $f10a
x:    addt $f10a, $f8, $f12
y:    addt  $f4, $f6,  $f4
z:    addt  $f4, $f8, $f10
```



With thanks to CS370, *Superscalar Processing* at CMU

Pentium Pro (P6)

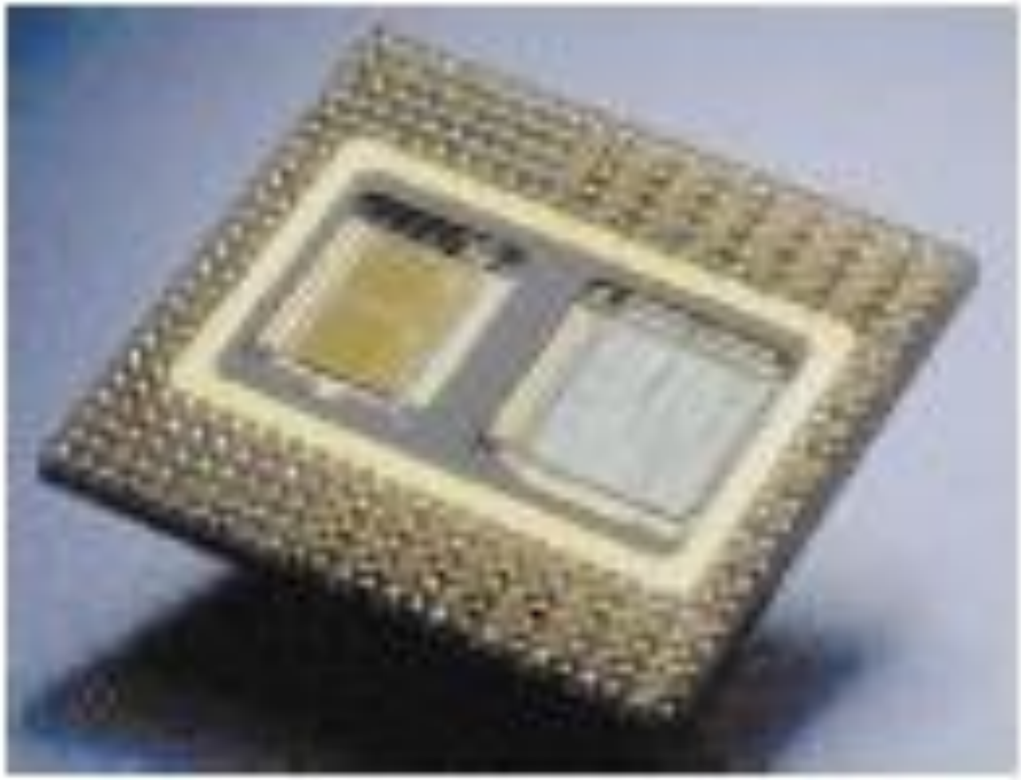
- History
  - Concept work on Dynamic Execution engine started 1990
  - Announced in Feb. ’95
  - Initially 133MHz, 2.9V
- Features
  - Dynamically translates instructions to more regular format
    - Very wide RISC instructions
  - Executes operations in parallel
    - Up to 5 at once
  - Very deep pipeline
    - 12–18 cycle latency
  - 5.5M transistors, 0.6 micron.

With thanks to CS370, *Superscalar Processing* at CMU

What is Dynamic Execution?

- Multiple Branch prediction:
  - First, the processor looks multiple steps ahead in the software and predicts which branches, or groups of instructions, are likely to be processed next. This increases the amount of work fed to the processor.
- Dataflow analysis:
  - Next, the P6 analyzes which instructions are dependent on each other.s results, or data, to create an optimized schedule of instructions.
- Speculative Execution:
  - Instructions are then carried out speculatively based on this optimized schedule, keeping all the chip’s superscalar processing power busy, and boosting overall software performance.

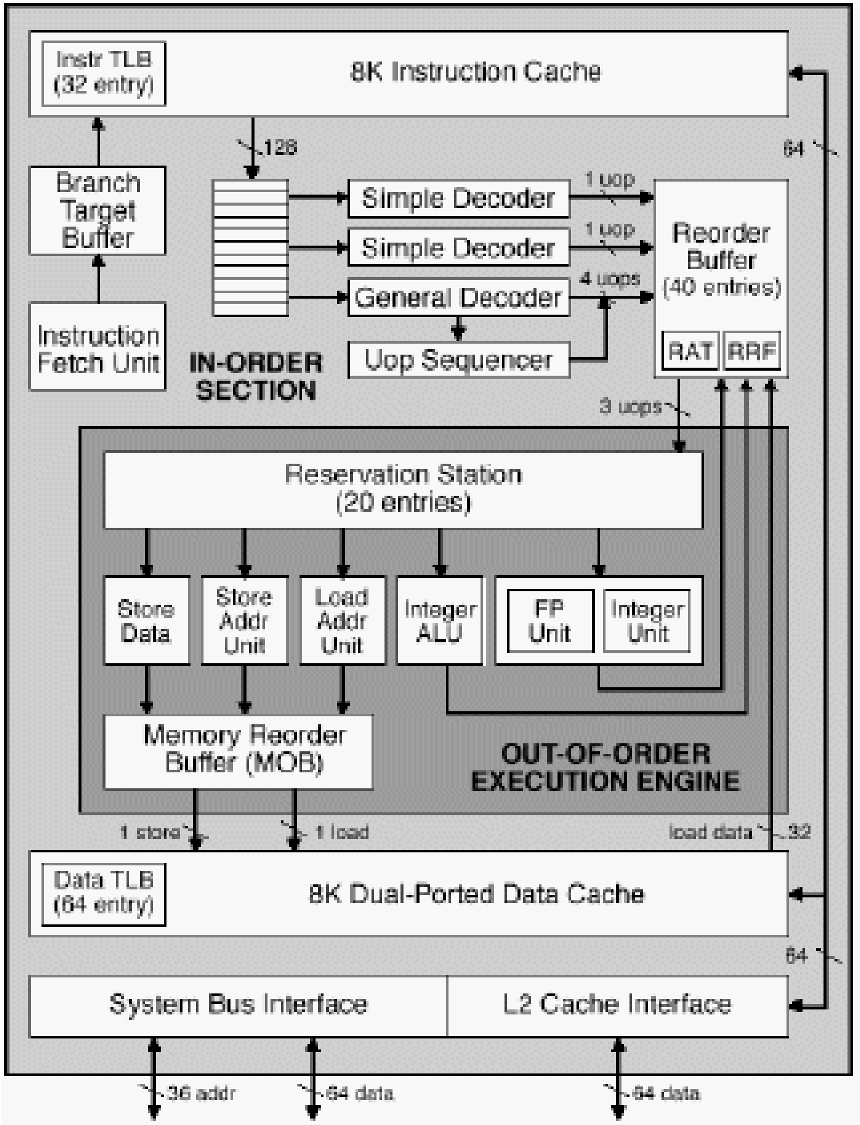
Pentium Pro Packaging



Early examples used an expensive two-die-in-a-package construction

With thanks to CS370, *Superscalar Processing* at CMU

Pentium Pro Block Diagram



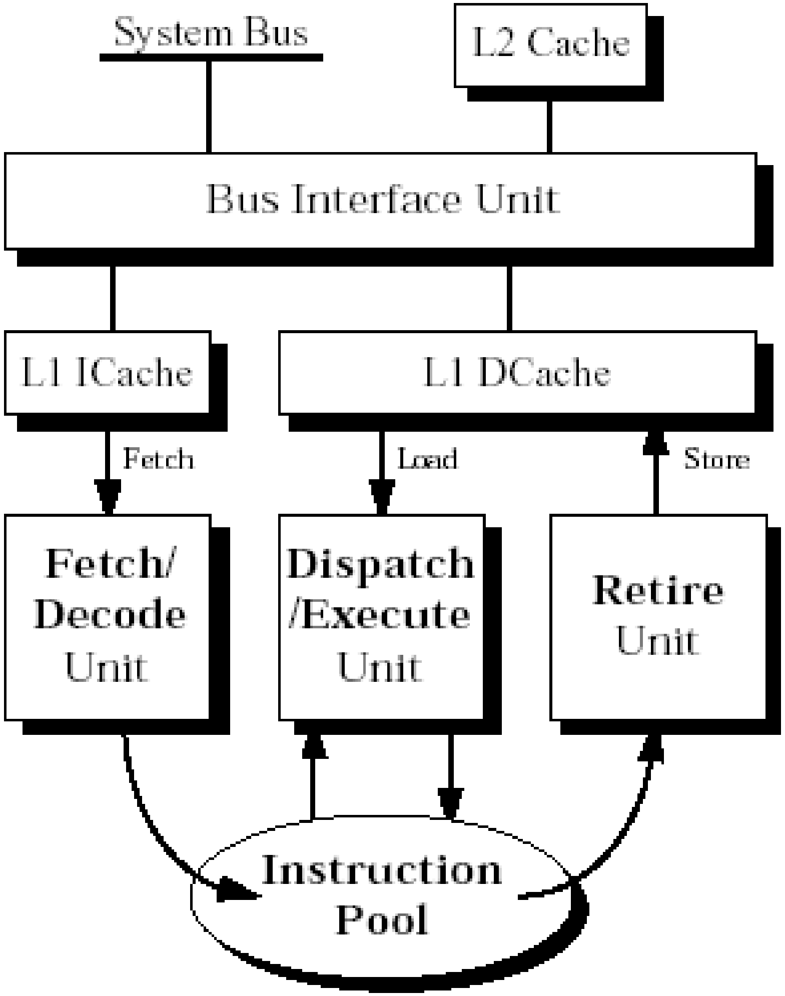
Microprocessor Report
2/16/95

Pentium Pro 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

With thanks to CS370, *Superscalar Processing* at CMU

Simplified architecture



With thanks to CS370, *Superscalar Processing* at CMU

In-order and Out-of-order units

- The FETCH/DECODE unit: An in-order unit that takes as input the user program instruction stream from the instruction cache, and decodes them into a series of micro-operations (uops) that represent the dataflow of that instruction stream. The program pre-fetch is itself speculative.
- The DISPATCH/EXECUTE unit: An out-of-order unit that accepts the dataflow stream, schedules execution of the uops subject to data dependencies and resource availability and temporarily stores the results of these speculative executions.
- The RETIRE unit: An in-order unit that knows how and when to commit (“retire”) the temporary, speculative results to permanent architectural state.
- The BUS INTERFACE unit: A partially ordered unit responsible for connecting the three internal units to the real world. The bus interface unit communicates directly with the L2 cache supporting up to four concurrent cache accesses. The bus interface unit also controls a transaction bus, with MESI snooping protocol, to system memory.

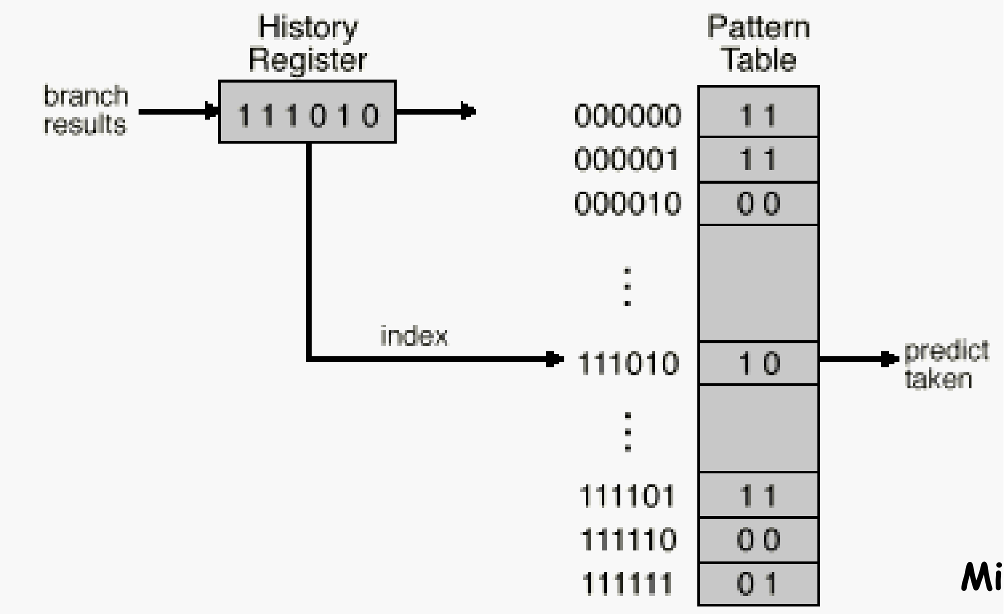
With thanks to CS370, *Superscalar Processing* at CMU

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

With thanks to CS370, *Superscalar Processing* at CMU

P6 Branch Prediction

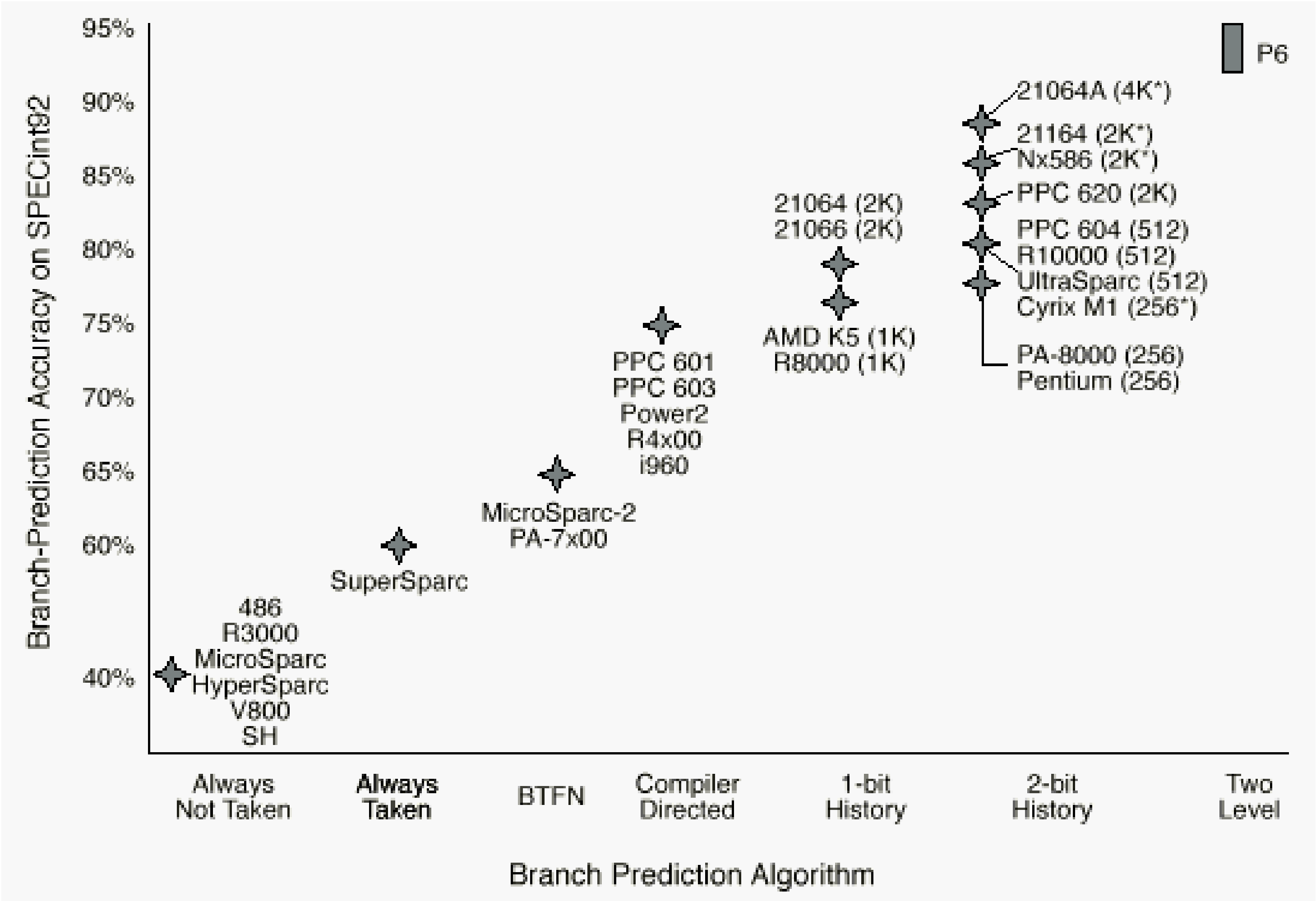


Microprocessor Report
March 27, 1995

- Two-Level Scheme
  - Yeh & Patt, ISCA ‘93
  - Keep shift register showing past *k* outcomes for branch
  - Use to index 2^k entry table
  - Each entry provides 2-bit, saturating counter predictor
  - Very effective for any deterministic branching pattern

With thanks to CS370, *Superscalar Processing* at CMU

Branch Prediction Comparisons



Microprocessor Report March 27, 1995

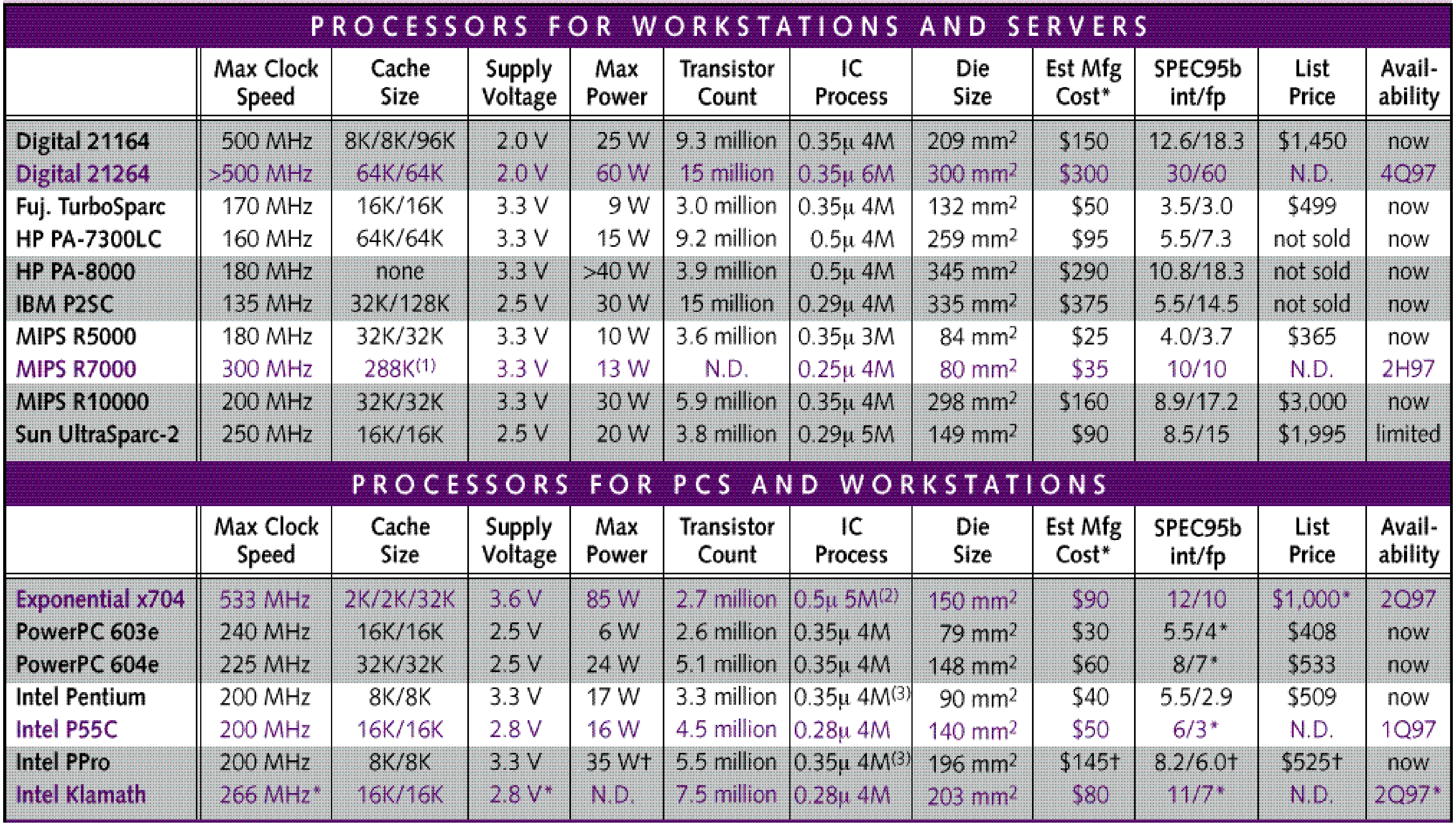
With thanks to CS370, *Superscalar Processing* at CMU

Limitations of x86 Instruction Set

- Not enough registers
  - too many memory references
- Intel has switched to a new instruction set for Itanium
    - IA-64, joint with HP
    - Will dynamically translate existing x86 binaries

With thanks to CS370, *Superscalar Processing* at CMU

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 mm^2	$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 mm^2	$300	30/60	N.D.	4Q97
Fuj. TurboSparc	170 MHz	16K/16K	3.3 V	9 W	3.0 million	0.35µ 4M	132 mm^2	$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 mm^2	$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 mm^2	$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 mm^2	$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 mm^2	$25	4.0/3.7	$365	now
MIPS R7000	300 MHz	288K(1)	3.3 V	13 W	N.D.	0.25µ 4M	80 mm^2	$35	10/10	N.D.	2H97
MIPS R10000	200 MHz	32K/32K	3.3 V	30 W	5.9 million	0.35µ 4M	298 mm^2	$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 mm^2	$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 mm^2	$90	12/10	$1,000*	2Q97
PowerPC 603e	240 MHz	16K/16K	2.5 V	6 W	2.6 million	0.35µ 4M	79 mm^2	$30	5.5/4*	$408	now
PowerPC 604e	225 MHz	32K/32K	2.5 V	24 W	5.1 million	0.35µ 4M	148 mm^2	$60	8/7*	$533	now
Intel Pentium	200 MHz	8K/8K	3.3 V	17 W	3.3 million	0.35µ 4M(3)	90 mm^2	$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 mm^2	$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 mm^2	$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 mm^2	$80	11/7*	N.D.	2Q97*

Microprocessor Report 12/30/96

Challenges Ahead

- Diminishing Returns on Cost vs. Performance
  - Superscalar processors require instruction level parallelism
  - Many programs limited by sequential dependencies
- Finding New Sources of Parallelism
  - e.g., thread-level parallelism
- Getting Design Correct Difficult
  - Verification team larger than design team
  - Devise tests for interactions between concurrent instructions
    - May be 80 executing at once

With thanks to CS370, *Superscalar Processing* at CMU

New Era for Performance Optimization

- Data Resources are Free and Fast
  - Plenty of computational units
  - Most programs have poor utilization
- Unexpected Changes in Control Flow Expensive
  - Kill everything downstream when mispredict
  - Even if will execute in near future where branches reconverge
- Think Parallel
  - Try to get lots of things going at once
- Not a Truly Parallel Machine
  - Bounded resources
  - Access from limited code window

With thanks to CS370, *Superscalar Processing* at CMU