ECSE-426

Microprocessor Systems
Winter 2015

RISC vs CISC

- Reduced Instruction Set Computing
  - Move complexity from hardware to software
  - Reduce CPI, increase code size
  - Simple, single-cycle instructions to perform basic functions
  - Assembler instructions (often) = microcode instructions
  - Simple addressing modes; use pipelining to lower CPI.

RISC vs CISC

- Reduced Instruction Set Computing
  - Move complexity from hardware to software
  - Reduce CPI, increase code size
  - Simple, single-cycle instructions to perform basic functions
  - Assembler instructions (often) = microcode instructions
  - Simple addressing modes; use pipelining to lower CPI.
- Complex Instruction Set Computing
  - Move complexity from software to hardware
  - Increase CPI, decrease code size
  - Complex instructions
  - Memory-to-memory addressing; microcode control unit.

Post-RISC

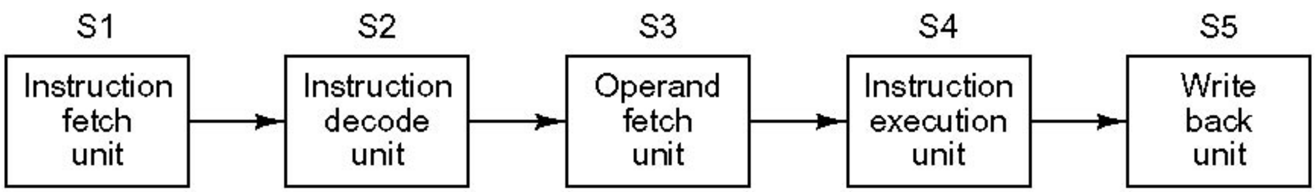
- Additional registers
- On-chip caches clocked as fast as processor
- Additional functional units – superscalar operation
- Additional non-RISC instructions
- On-chip support for floating-point operations

Post-RISC

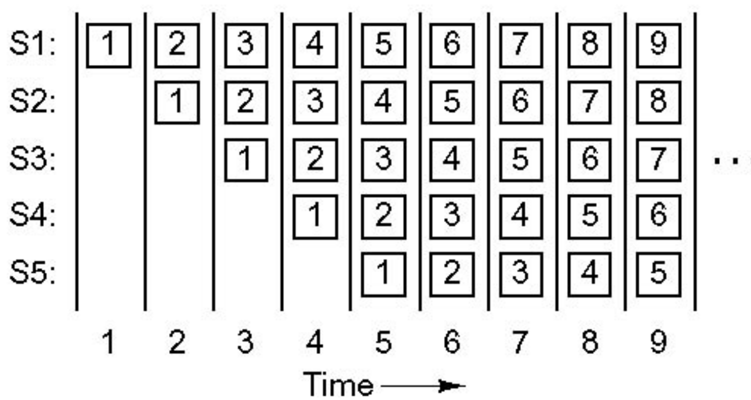
- Increased pipeline depth
- Branch prediction
- Out-of-order execution
- On-chip support for SIMD (single-instruction, multiple data) operations.

Basic Concepts - Pipelining

- Makes processor run at high clock rate
  - But might take more clock cycles
- Trick: overlap execution
  - Some overhead – first few instructions

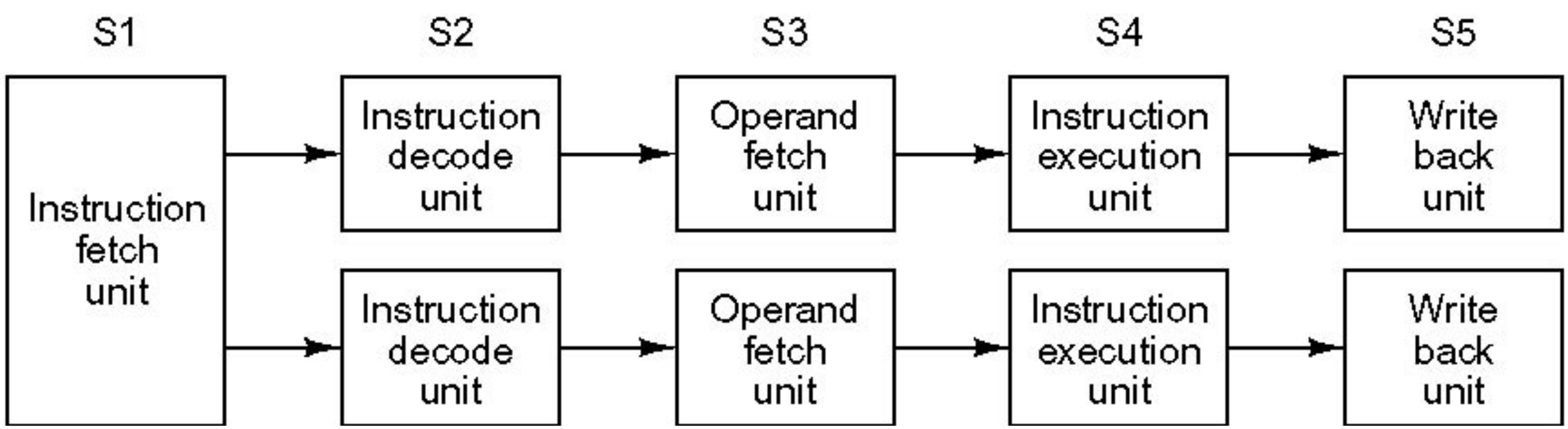


(a)



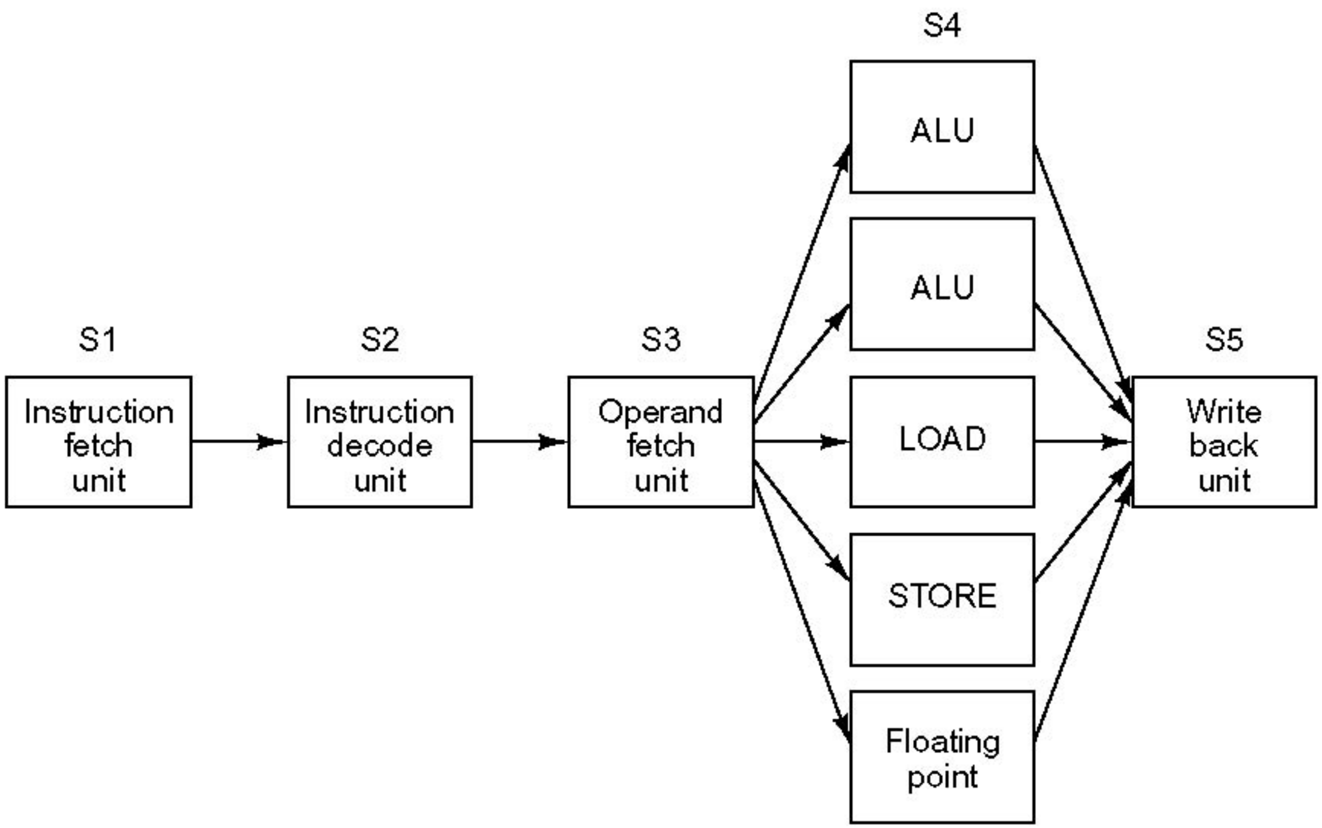
Other Speedups – Multiple Units

- Bottlenecks – execution in single pipeline units
  - ALU, especially floating point
- Resolution – provide multiple units



Superscalar Processors

- Common solution for modern processors
  - Multiple execution units



History of Pentium Architecture

- Discuss the architectural history of the most famous desktop processor
- In the process, highlight some microprocessor principles

The original Pentium

- Pentium Vitals Summary Table
  - Introduction date: March 22, 1993
    Process: 0.8 micron
    Transistor Count: 3.1 million
    Clock speed at introduction: 60 and 66 MHz
    Cache sizes: L1: 8K instruction, 8K data
    Features: MMX added in 1997

- A modest design even then (why?)

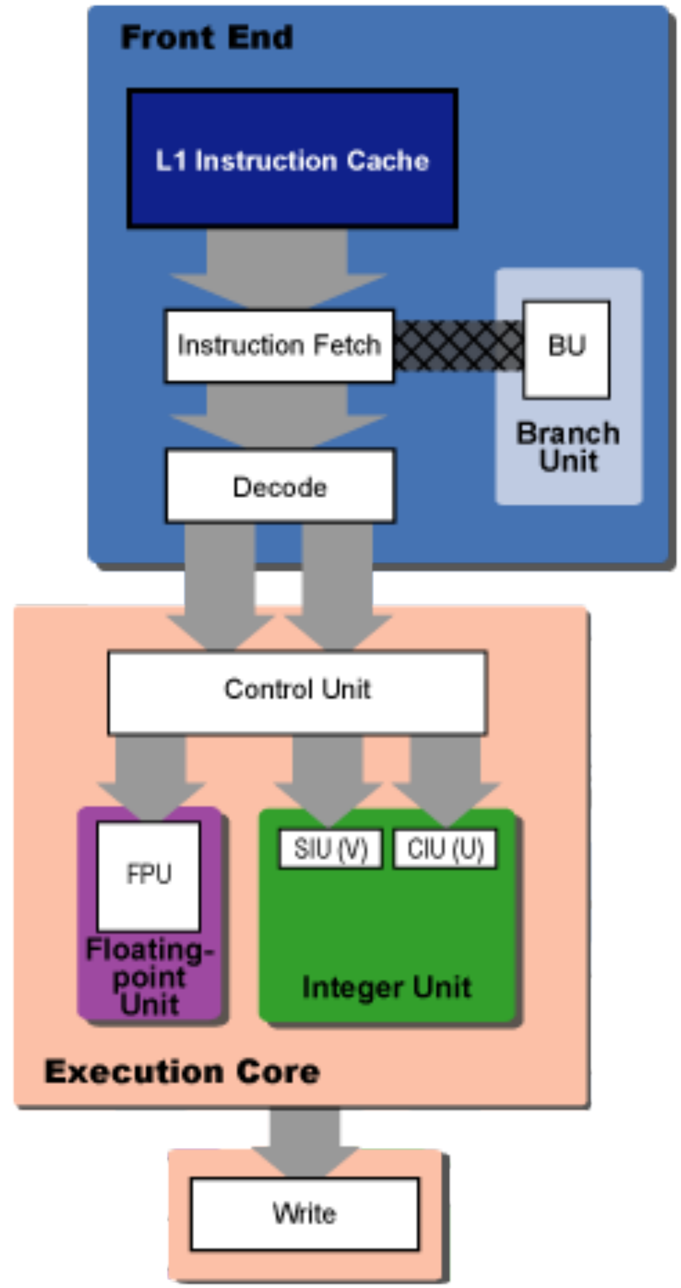
Backwards compatibility

- Intel’s first superscalar processor

- but
  - maintained x86 compatibility
  - critical strategic decision
  - entailed enormous sacrifices on performance, power consumption and cost.

Architecture

- Two-issue superscalar
- Two five-stage integer pipelines, U and V
- One six-stage floating point pipeline
- Front-end
  - some dynamic path prediction
  - most resources spent on backwards-compatibility



Integer unit issues

- Cannot get peak performance if you can’t keep code pumping through ALU
- Branch mispredicts, cache misses lead to pipeline bubbles
- Integer-intensive applications
  - often contain branch-intensive code that exhibits poor locality of reference

Integer Pipes in the Pentium

- U and V not fully symmetric
- U – default pipe, slightly more capable, contained shifter
- Pipelines not fully independent (some combinations of integers could not be processed simultaneously)
- Solid performance, but not outstanding

Integer unit principles

- x86 ISA – two operand instructions
  - add A, B -> A = A+B
  - inconvenient when you want to store result in C
- Two types of integer instructions
  - simple: ADD, SUB (little overhead, vast majority)
  - complex: multiplication, division (high overhead, small fraction of instructions)
- Decision: make the simple instructions fast

Floating point

- Floating point intensive applications
  - Games, simulations, 3D rendering, audio processing
  - most multimedia- and entertainment-oriented computing applications

- In some ways opposite of integer-intensive applications
  - Few branches, very predictable
  - Excellent locality of reference for instruction cache
  - Memory bandwidth important – need to stream large files from main memory

Floating point

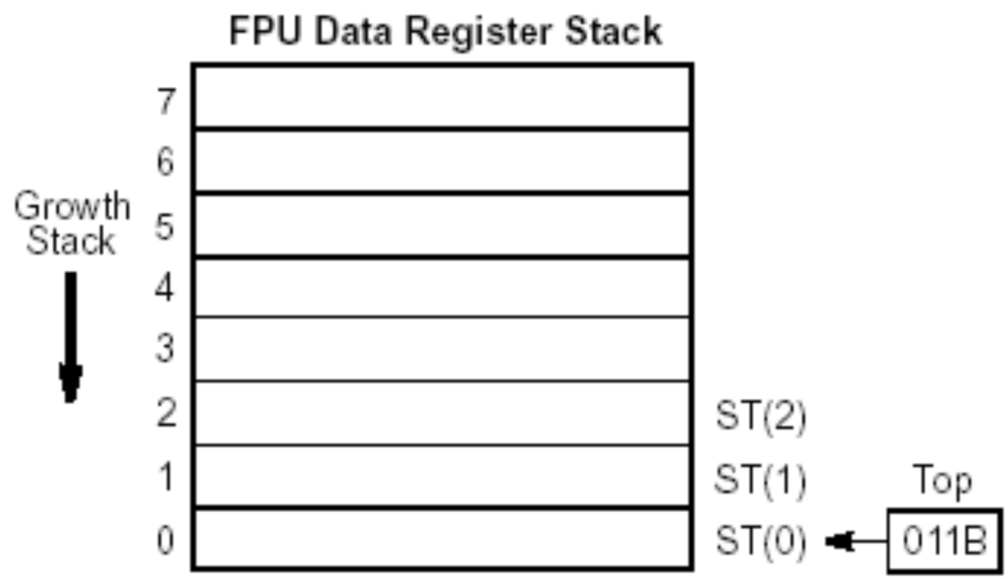
- Mediocre implementation
- not competitive with comparable RISC chips

- Could only issue a floating-point and an integer operation simultaneously under very restrictive circumstances

- x87 stack-based floating point architecture was poorly designed
- x87 low register count: only 8 architectural registers

x87 issues and quirks

- Two operand instruction format
- Stack-based register file
  - Eight 80-bit registers arranged in stack
  - Can be pushed and popped but also indexed
  - For every operation at least one operand must be in ST(0)



x87 issues and quirks (2)

- Example: if register A is in ST, then
  - FADD ST, B
  - -> ST = ST + B
  - A = A + B

- Problem?
  - Two operand limit AND stack-based constraint -> compilers can’t eliminate performance penalties
  - Extra command FXCH to move register to top of stack
  - Need microarchitectural hack in order to simulate a flat register file (Pentium III)

Pentium integer pipeline

- Prefetch/fetch
  - Fetch instructions from cache and align in prefetch buffers for decoding
- Decode1
  - Decode instructions into Pentium's internal instruction format + branch prediction
- Decode 2
  - Microcode ROM if necessary + address computations
- Execute (execute instruction)
- Write-back (write result to register file)

x86 overheads

- ~30 percent of transistors for x86 support
- Second decode stage
  - RISC ISAs have simple addressing modes
  - x86 has complex modes requiring extra addressing computations
- Pentium microcode ROM (decode complex instructions)

- Front-end:
  - x86 instructions not of uniform size so could straddle cache lines
  - Decode logic had to support segmented memory model
  - Dedicated address hardware needed 4 input ports

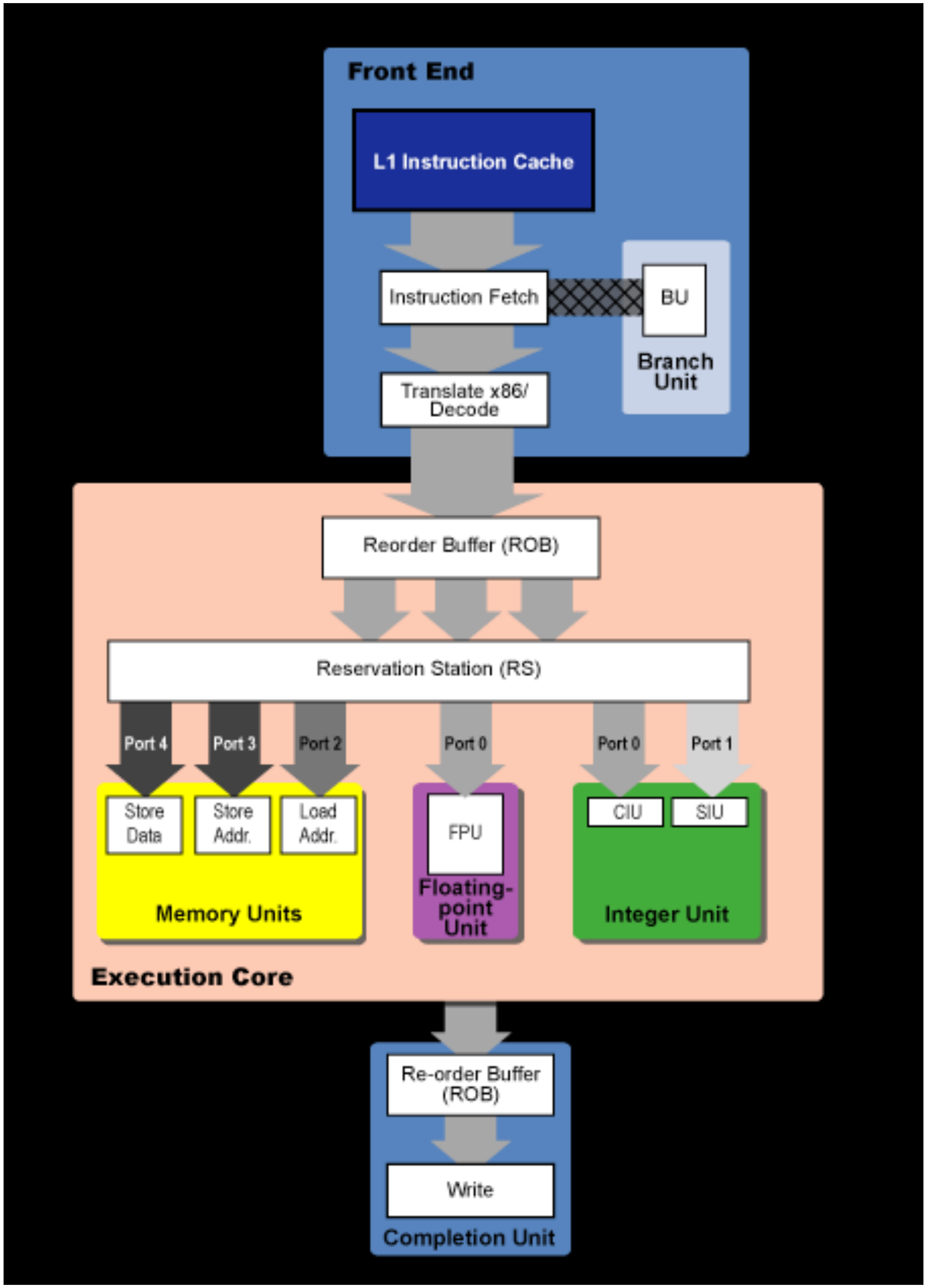
- Fortunately: legacy overhead largely fixed in size, transistors shrunk, so relative cost diminished (~10%)

P6 Architecture

	Pentium Pro	Pentium II	Pentium III
Intro date	Nov. 1995	May 1997	Feb. 1999
Process	0.6/0.35 mic	0.35 micron	0.25 micron
Transistor	5.5 million	7.5 million	9.5 million
Clock speed	150-200 MHz	233-300 MHz	450-500 MHz
L1 cache	8K instruction 8K data	16K instr 16K data	16K instr 16K data
L2 cache	256/512K on	512K (off die)	512K on-die
Features	No MMX	MMX	MMX, SSE

P6 Architecture

- Resounding success
- Most important change
  - Decoupling of front and back end
  - Front end: fetching/ decoding
  - Back end: execution
- Instruction window



Decoupling

- Pentium
  - Instructions travelled directly from decoding hardware to execution hardware
  - Hardwired rules dictating which instructions could go to which execution units
  - Problem: superscalar processor tries to do parallel execution, but static rules-based approach does not adapt to code stream
  - Two issue machine (only looks at 2 instructions at a time to see if they can be executed simultaneously)

Decoupling – reservation station

- Place decoded instructions in buffer
- Issue them whenever they're ready to be executed – not just in parallel but perhaps out of order.
- Each instruction has execution requirements
  - Needs input from an unexecuted instruction
  - Waiting for busy execution unit
- Wait in buffer until execution requirements are met
- Up to 3 instructions per cycle from decoders to reservation station
- Up to 5 instructions per cycle from reservation station to execution units

Reorder buffer (ROB)

- ROB first records all essential information about each instruction entering core (40 available entries)
- Primary function: ensure finished instructions get put back in program order & results written to register file in correct sequence
- After execution, results go back to ROB.
- Final write-back (retirement) can only occur when all previous instructions have been retired.

Instruction Window

- RS + ROB combination -> instruction window
- ROB: can track 40 instructions in various stages of execution
- RS: can hold and examine up to 20 instructions to determine optimal order
- Like Tetris
  - But 3 pieces at a time
  - And see a window of 20 pieces

Register renaming

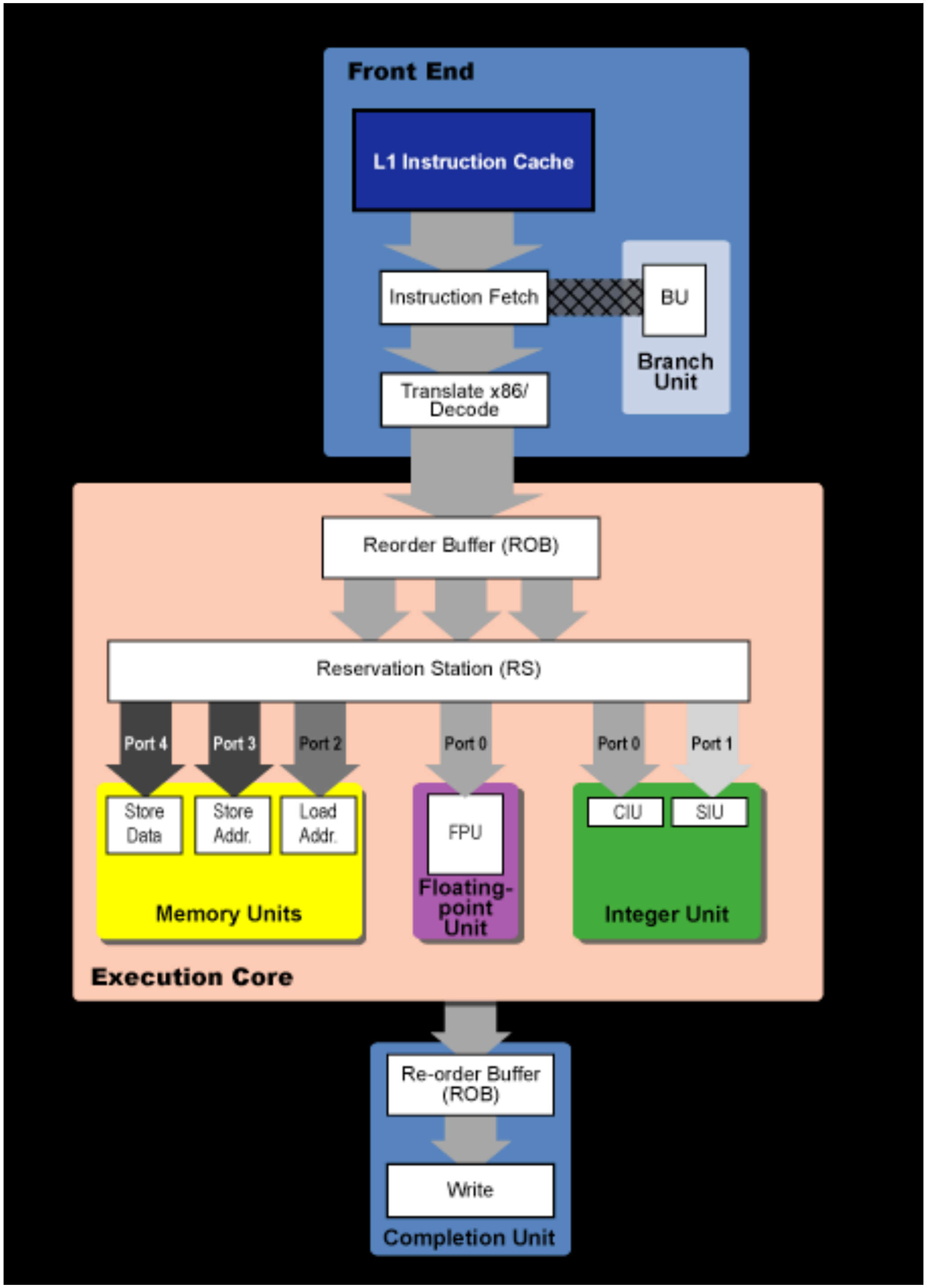
- x86 has only 8 general purpose registers and 8 floating point registers (PowerPC ISA has 32 of each)
- Register renaming allows the processor to use more registers (40)
  - some tricks involved to make the program think it is only using 8
- Each of the 40 core ROB entries has a data field
  - Holds program data just like register
  - Gives execution core 40 micro-architectural registers
  - Register Allocation Table (RAT) implements renaming

P6 execution core

- 2 integer ALUs + floating point unit
- 3 execution units solely for memory access
  - Load address, store address and store data
- Integer ALUs
  - single-cycle throughput and latency for most operations
  - Multiplies and divides have single-cycle throughput but 4 stage latency
- Floating point unit
  - FXCH instruction now a register rename in ROB (so effectively zero cycles to execute)
  - Compilers can use this to simulate a flat register file

Execution core

- Five issue ports from reservation station
- 5 instructions per cycle
- Complex integer unit and floating-point unit share a port.



P6 Pipeline (12 stages)

- Branch Target Buffer access + instruction fetch
  - First 3.5 stages
  - 2 cycle instruction fetch stage (keeps the L1 cache access latency from holding back clock speed of the processor as a whole)
- Decode (2.5 stages)
  - x86 instructions to P6 internal, RISC-like format
- Register rename (1 stage)
- Write to RS (1 stage)
- Read from RS (1 stage)
- Execute (1 cycle or multiple cycles)
- Retire (2 final cycles for writing back results to ROB and register file)

Pipeline + Branch Prediction

- Why lengthen the pipeline?
  - Crank up clock speed
  - Hide hiccups in the fetch/decode stages
    - First nine stages are a deep buffer for instructions
- Dynamic branch prediction accuracy improved from ~75% to above 90%
- More important for longer pipelines (pipeline flush means more lost cycles – 3 vs 9)

The x86 Legacy

- Extra time in decode phase, but extra 1.5 cycles go to ISA translation
- Offset by increase in length of pipeline
- Heavy reliance on register renaming

- Many transistors for P6 decoding logic and ISA translation
  - 2 simple/fast decoders + 1 complex/slow
  - Estimates: close to 40 % of transistor budget
- Cost was high for Pentium Pro
  - Only a 16K L1 Cache because of transistor shortage
  - Comparable RISC processors – 32/64K L1 cache

P6 Core - Incarnations

- Pentium Pro
  - Short on transistors/cache/features
  - Even MMX was jettisoned from the Pentium
  - But considerable improvement –> commodity server market
- Pentium II
  - On-die, split L1 cache doubled in size
  - Single-edge cartridge (daughtercard with 256K L2 cache) connected with fast backside bus
  - Two new MMX units (integer only)

P6 Core – Incarnations (2)

- Pentium III
  - Gigahertz race with AMD
- Race slowed down afterwards
- But had major ramifications on industry