System Architecture

PC Architecture
Part 1

Ashish Kuvelkar

HPC Technologies Group

C-DAC, Pune

Why we are studying this?

- PC architecture is one of the most popular architectures
- Open architecture: details are available in public domain
- Will get to know
  - Evolution of x86 architecture
  - Evolution of system architecture
    - Memories
    - Chipsets
    - Busses

What will be covered

- History of Computing
- IBM PC Architecture
- Generations of CPUs used in PC
- Fundamental principles of microprocessor design at micro-architecture level

Prerequisites

- Knowledge of
  - Microprocessor fundamentals
  - Microprocessor based system design
    - Various components
  - Various types of memories
  - Assembly language programming

General Guidelines

- Slides will be shared, no need to write down the contents
- Not understood something, feel free to ask questions…
- … to the teacher, not to your neighbor
- Put in good efforts to apply theory to lab assignment
- Time is limited, complete given task within stipulated schedule

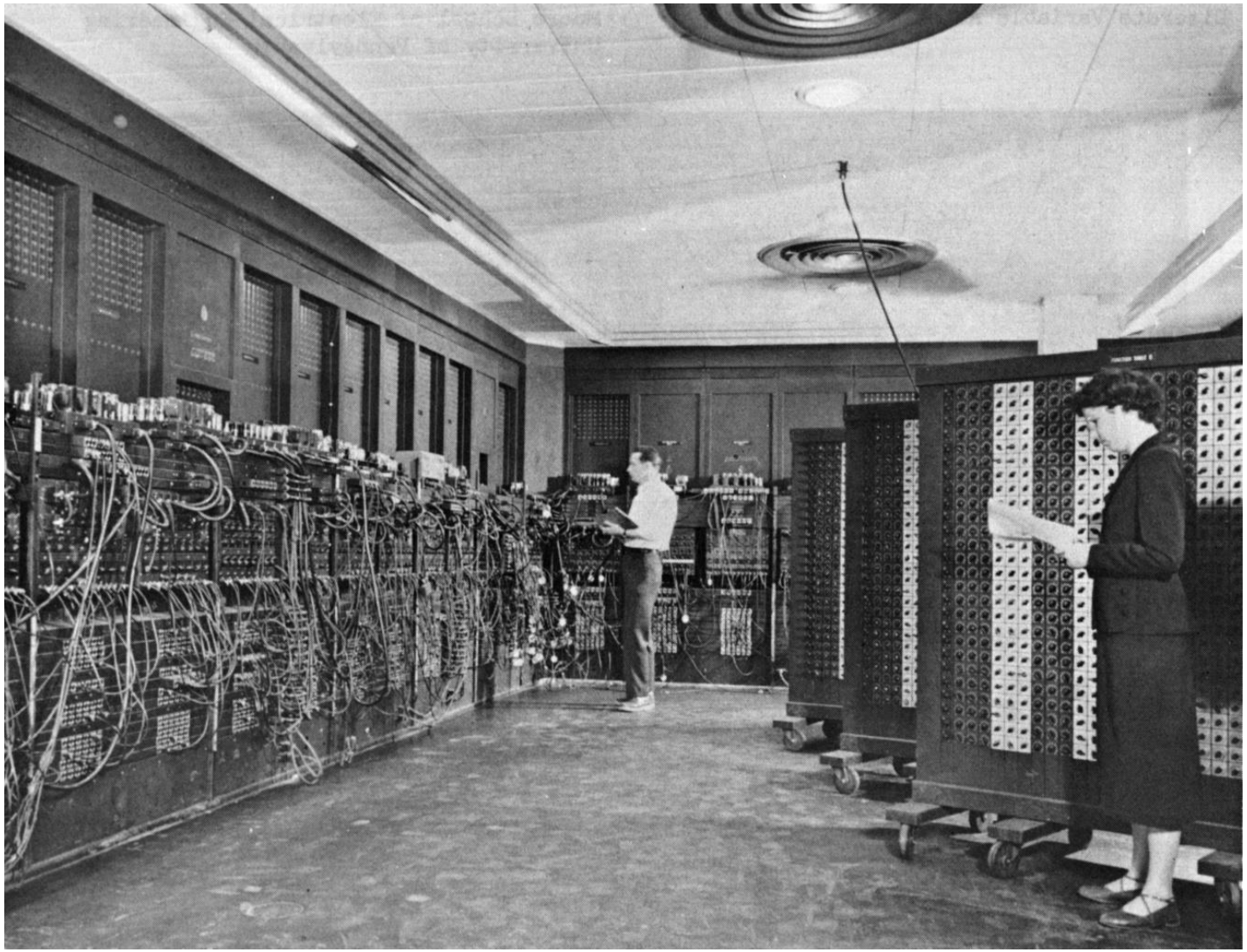
Introduction

Generations of Computers

- Generation is determined by
  - Device technology
  - System architecture
  - Processing mode
  - Languages used

Generations of Computers

- First Generation
  - Electromechanical relays used as switching devices in 1940s and vacuum tubes in 1950s
  - CPU structure was bit-serial
  - Machine language for programming
  - ENIAC and UNIVAC

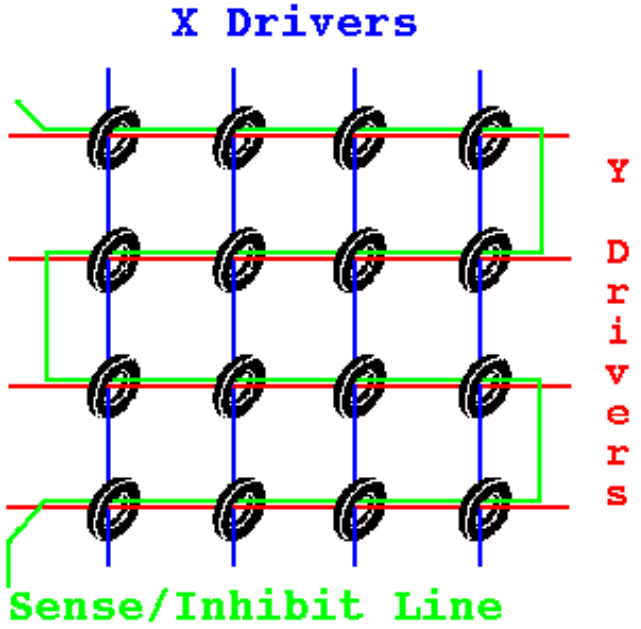


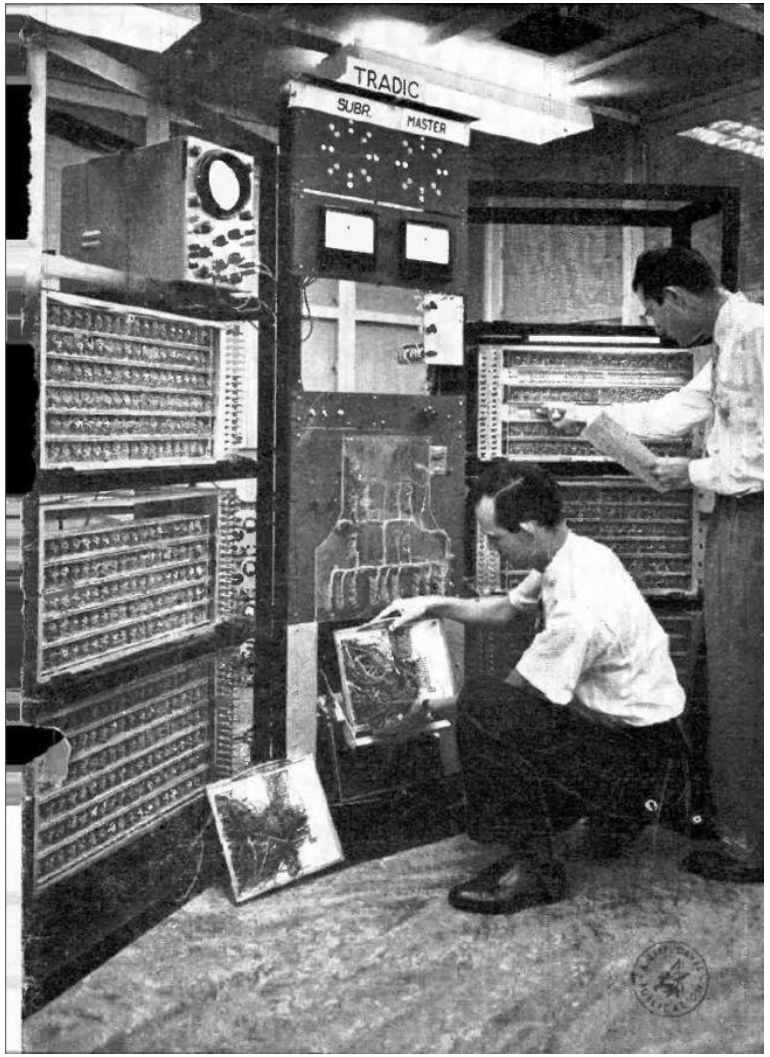
ENIAC

UNIVAC

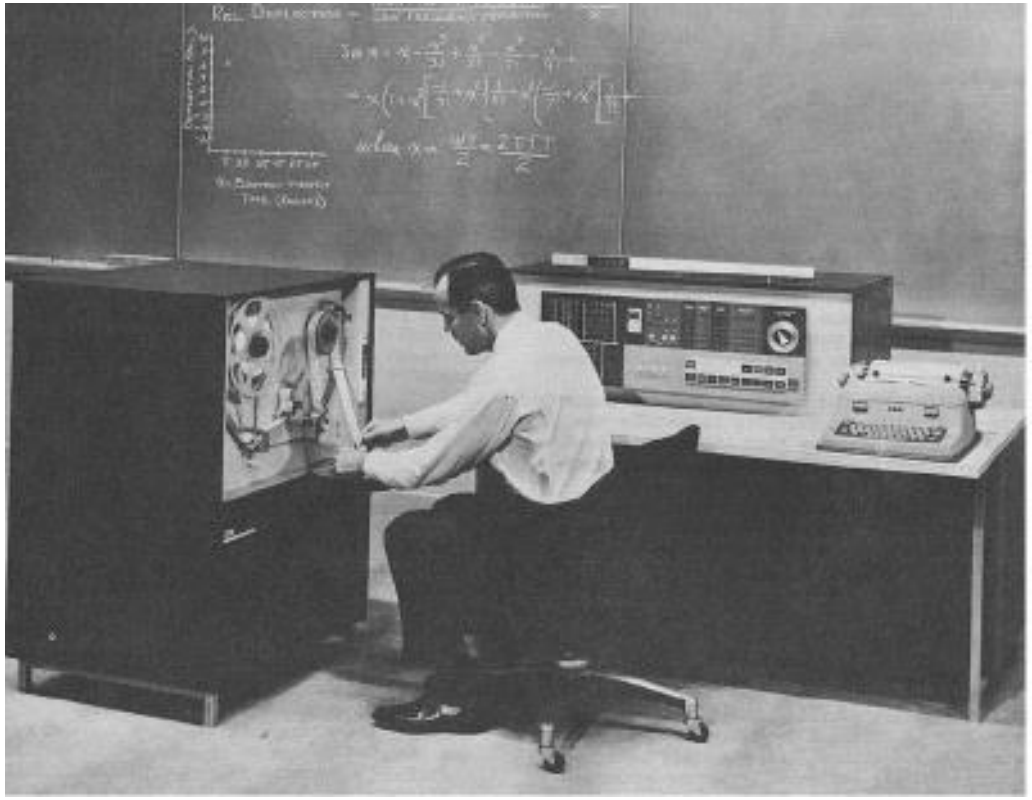
Generations of Computers

- Second Generation
  - Transistors replaced vacuum tubes as switching devices
  - Core memory was used as storage device
  - Assembly language, COBOL and FORTAN were used for programming
  - TRADIC (by Bell Labs) and IBM 1620





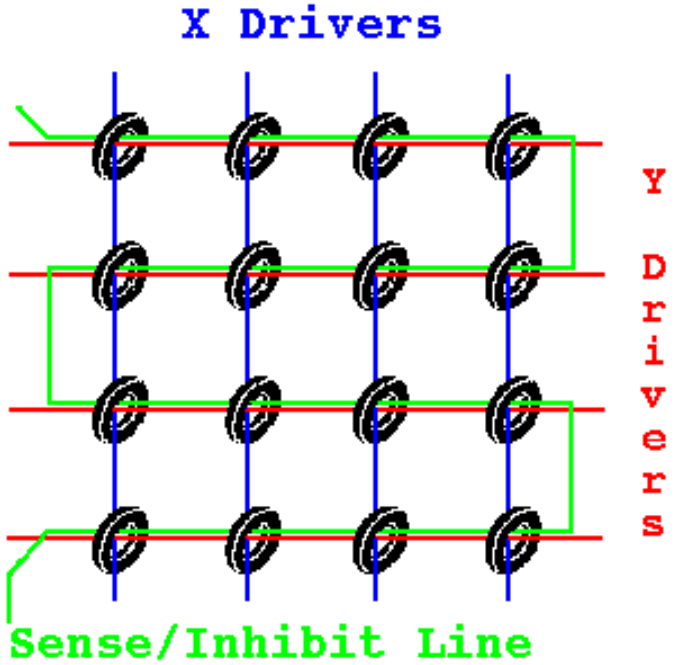
TRADIC



IBM 1620

Generations of Computers

- Third Generation
  - LSI and MSI circuits were used as building blocks
  - Core memory was still used as storage device
  - Operating systems allowed multiple application programs to be run simultaneously
  - CDC-6600 and IBM 360 series



CDC-6600

IBM-360

Generations of Computers

- Fourth Generation
  - VLSI circuits and Microprocessors are used as building blocks
  - Solid state memories are used as storage device
  - Operating systems became GUI based and mouse became the input device
  - Cray-1, Cray X-MP, IBM 370, Macintosh

CRAY-1

IBM-370

A brief History of Computing

"I think there is a world market for maybe five computers"

- Thomas Watson, chairman of IBM in 1943

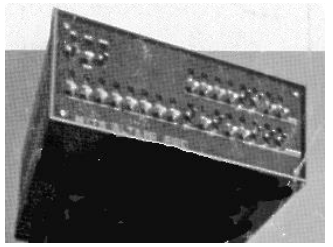
ENIAC (Electronic Numerical Integrator and Computer): deployed, at the Ballistic Research Laboratory, USA in 1946.

Weight was 30 tonnes, contained 18,000 Electronic Valves, consumed 25KW of electrical power,could do around 100,000 calculations a second

UNIVAC-I: The first commercially successful electronic computer launched in 1951

History of PC

- Altair shipped the first “PC” in 1975
  - I/O through front panel switch and LEDs
  - Primarily for hobbyists and hackers
- Apple – II was the first real PC
  - I/O though keyboard and color graphics
  - Targeted at home and business market
- Other vendors included Tandy, Commodore & TI
  - Each vendor had his own architecture, bus, OS.
  - By purchase, implicit commitment was made to that vendor’s standard



IBM PC Architecture

Success of PC architecture

- PC architecture dates back to 1978
  - IBM launched PC in August 1981
- “Personal” computing was introduced first time
  - Earlier ones were Mainframes and Minis
- “Open design” proved to be a prime factor for success
  - Design documentation
    - Hardware Schematics
    - Specifications of I/O bus
    - BIOS listing, and BIOS calls standardization
- Adopted by multiple vendors
- Compare this with Apple’s Macintosh systems

Quotable Quotes

- "There is no reason anyone would want a computer in their home." Ken Olson, president, chairman and founder of Digital Equipment Corp. in 1977

- "640k ought to be enough for anybody.", Bill Gates in 1981

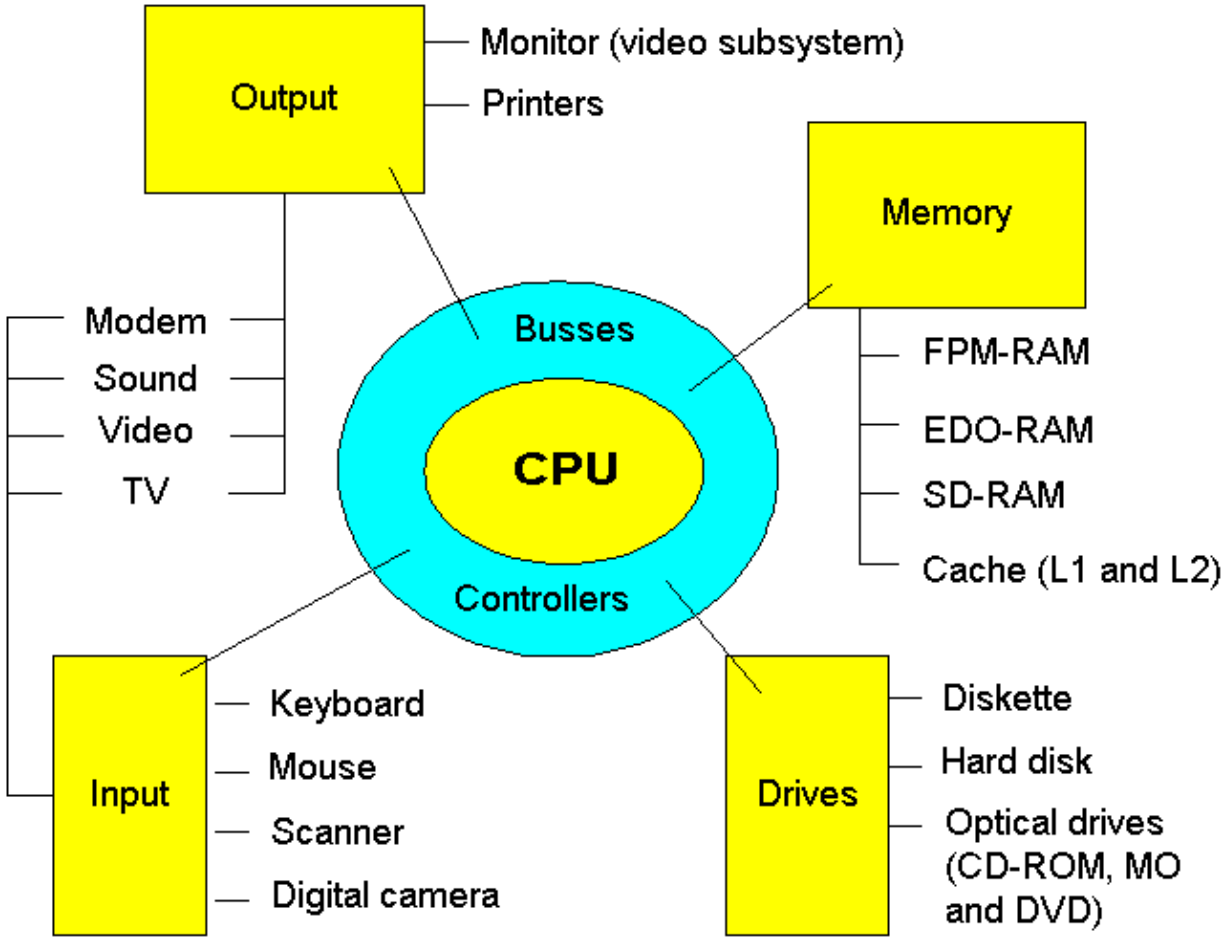
Specifications of IBM PC

- CPU – 8088 @ 4.77 Mhz
- Optional Co-processor 8087
- Main Memory 16 to 64kb, expandable to 640kb
- Two 5 ¼” (160Kb) FDD
- Display adapter CGA
  - 320X200X4
- 84 key keyboard
- 14” display
- MS-DOS 1.0

PC architecture

- Standard interface in hardware
  - Architecture of system (PC) is different from architecture of microprocessor (8088)
  - Firmware provides abstraction layer for the system software (OS)
- Software: adaptation of a standard API, used by future hardware
  - e.g. windows API framework
- What about peripherals ?
  - standardization follows popularity

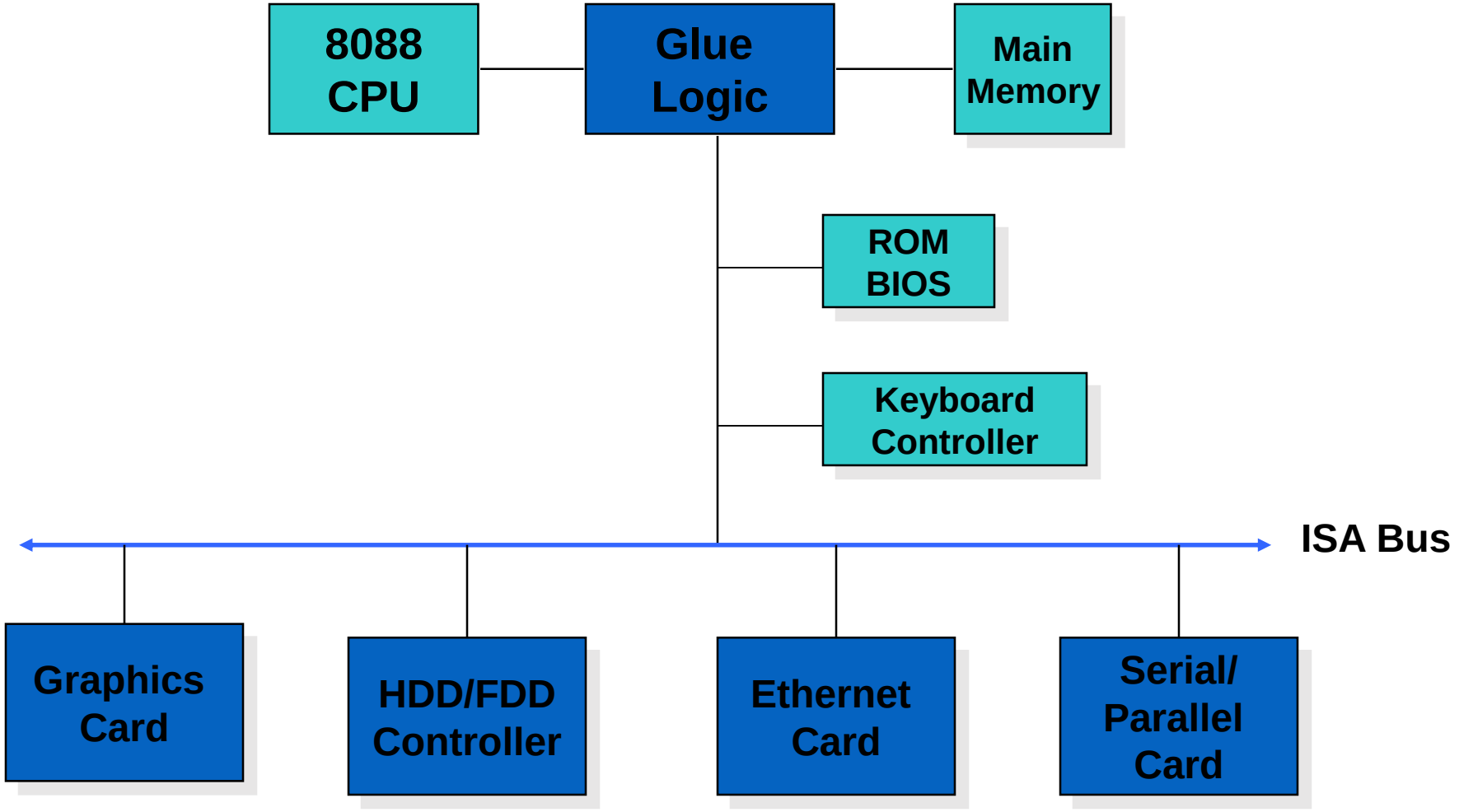
Input Output systems of PC



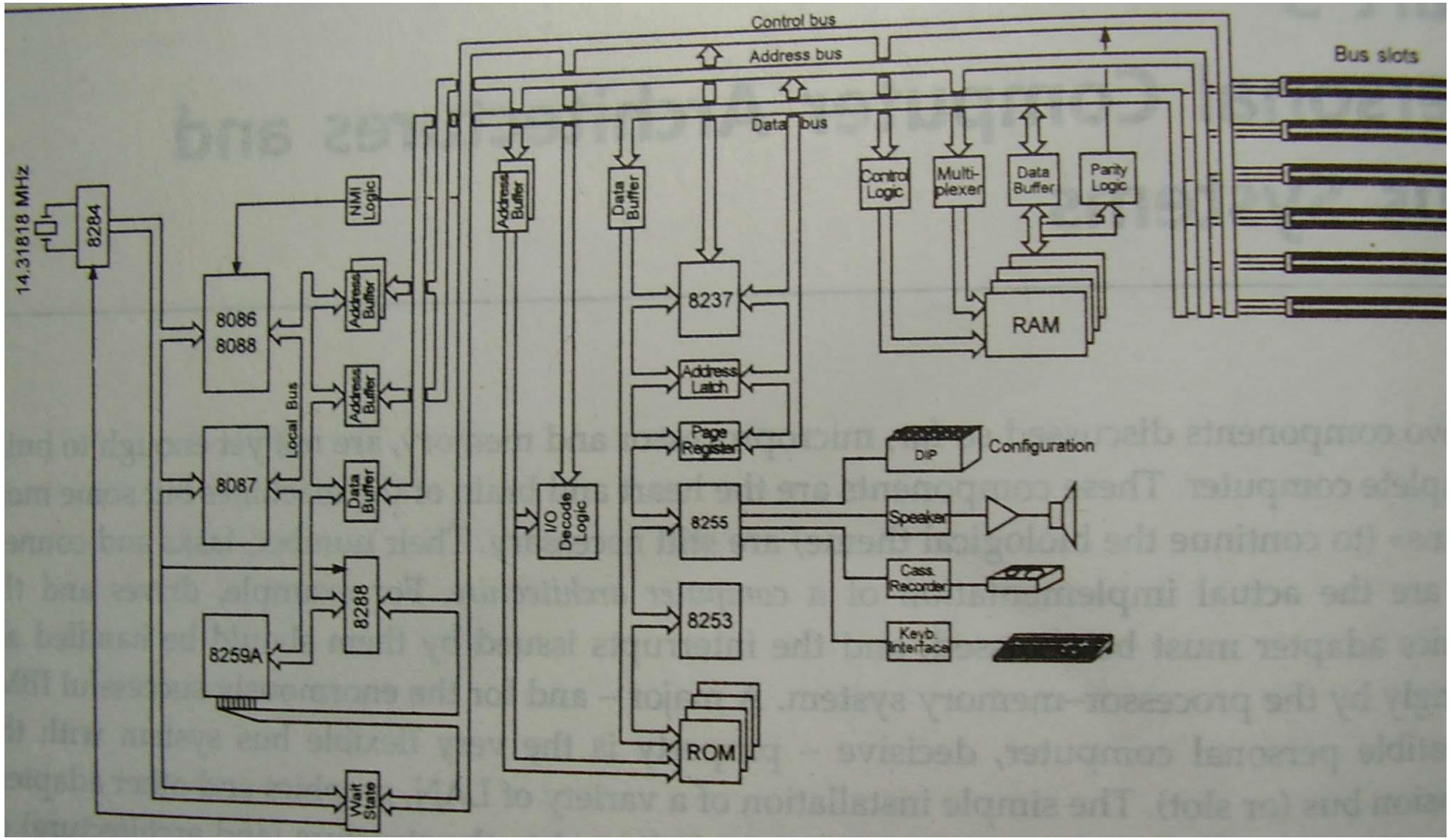
Main components of a PC

- CPU
- Memory subsystem
- I/O subsystem
- A controller which binds these blocks together.

PC XT Block Diagram



XT block diagram



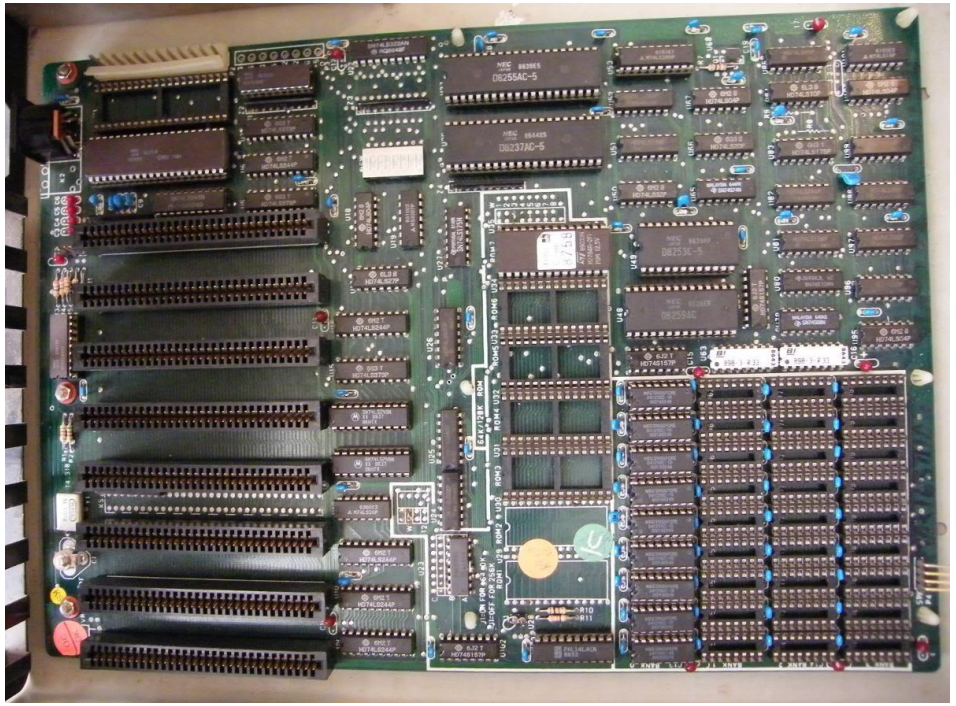
The block diagram

Major components:

- CPU, cache, main memory
- System controller
- I/O subsystem: on board devices + bus slots
- Boot EPROM/FLASH, RTC, serial/parallel, keyboard
- FDD/HDD controllers
- Power supply

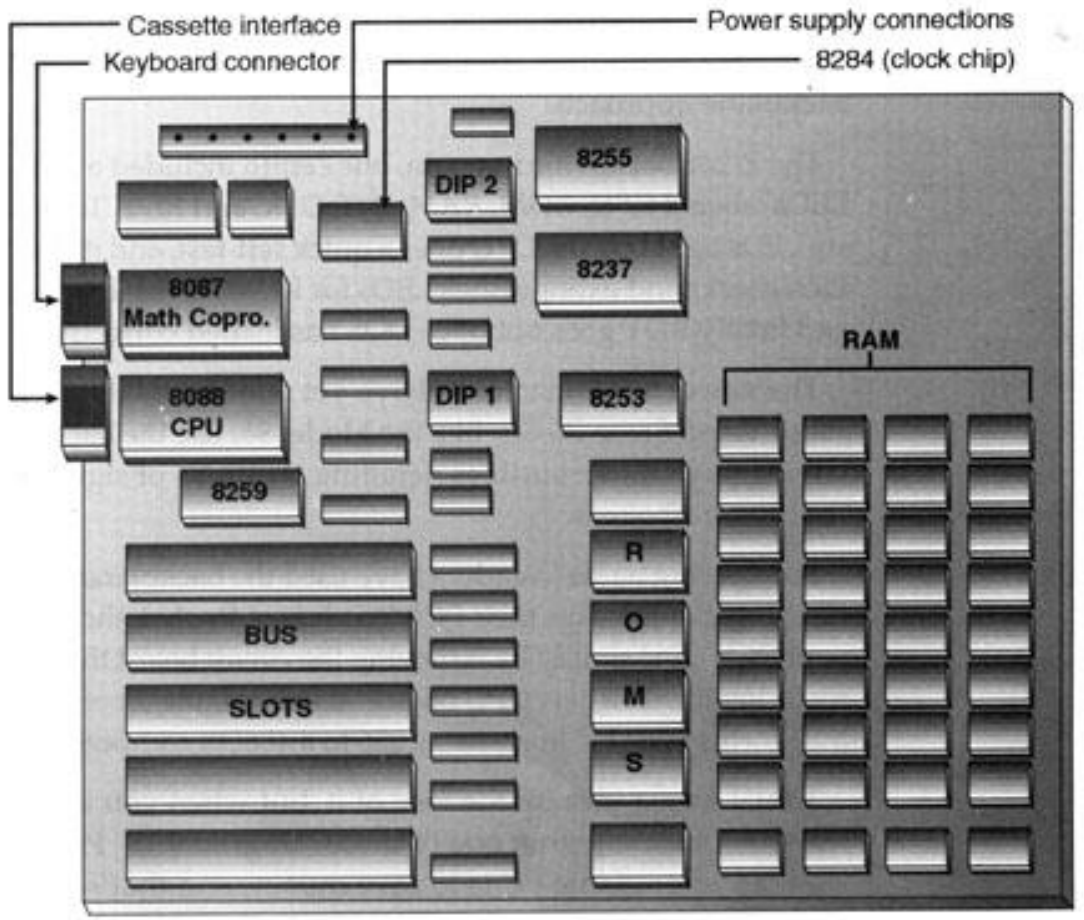
Typical system based on 8088

- 8088 CPU
- clock generator/ controller
- DMA controller
- interrupt controller
- UART for communication
- Keyboard controller
- 8-bit wide memory
- LSI/MSI blocks: Decoders, transceivers, multiplexers

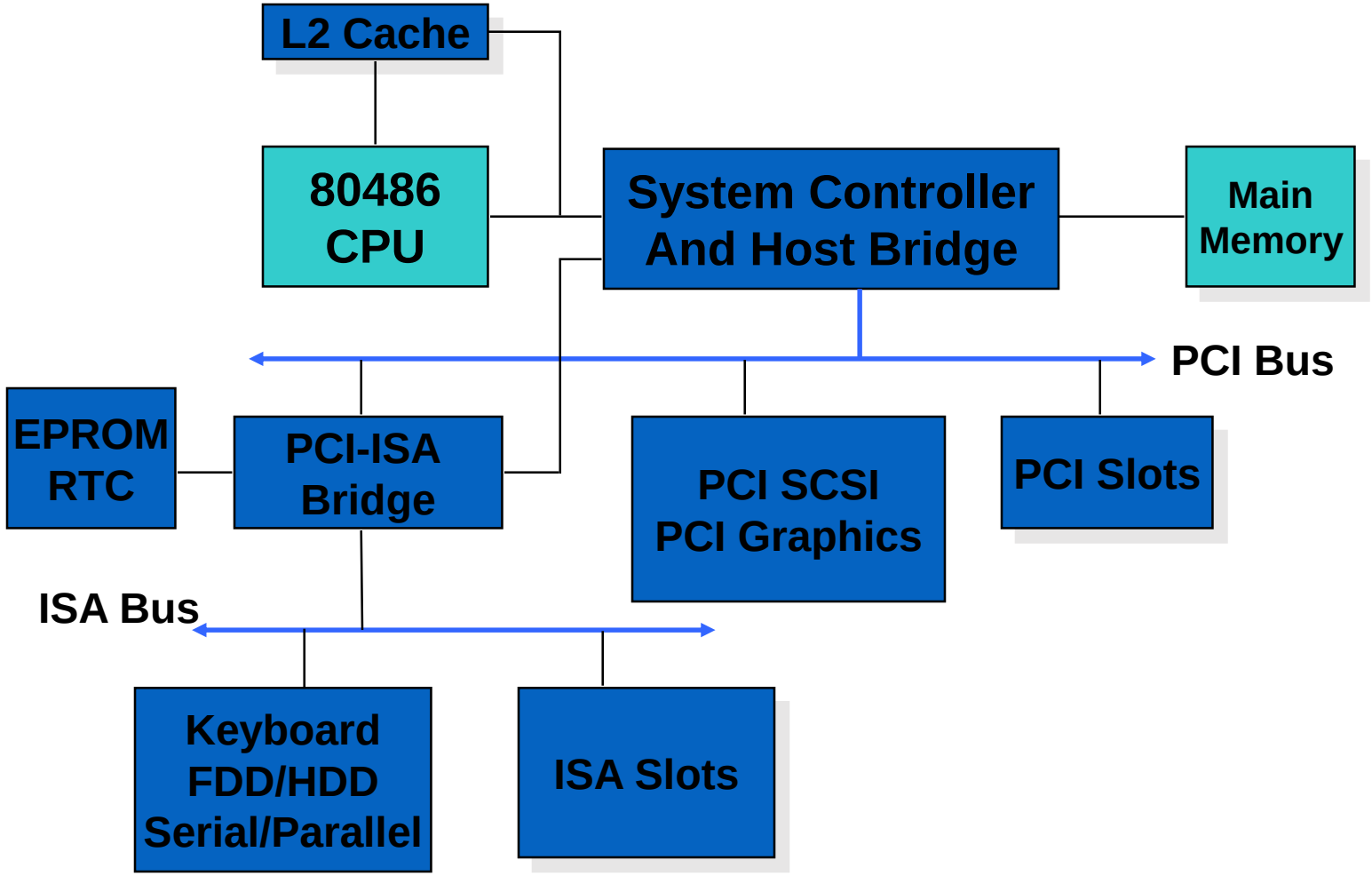


Typical system based on 8088

FIGURE 3.1
IBM PC system board outline



PC AT Block diagram



PC IO Map

- 000-00F DMA Chip 8237A-5
- 020-021 Interrupt 8259A
- 040-043 Timer 8253-5
- 060-063 PPI 8255A-5
- 080-083 DMA Page Register
- 2F8-2FF Asynchronous Communications(Secondary)
- 300-31F Prototype Card
- 320-32F Fixed Disk
- 378-37F Parallel Printer
- 3B0-3BF IBM Monochrome Display/Printer
- 3F0-3F7 Diskette
- 3F8-3FF Asynchronous Communications(Primary)

PC (AT) Interrupt usage

IRQ(8259)	Function
0	System timer IC
1	Keyboard controller IC
2	Second IRQ controller IC
3	Serial port 2
4	Serial port 1
5	Parallel port 2
6	Floppy disk controller
7	Parallel port 1

Note: IRQ 0 is assigned to interrupt 08 in vector table

PC AT Interrupt usage

IRQ (8259)	Function
8	Real time clock or RTC
9	Unused (redirected to IRQ 2)
10	general adapter use (USB)
11	general adapter use
12	M/B mouse port or general adapter use
13	Math coprocessor
14	Primary hard disk controller
15	Sec. HD cont. or general adapter use

Note: IRQ 8 is assigned to interrupt 70 in vector table

Processor Interrupt usage

- Interrupt Function
- 00h Divide By Zero
- 01h Single Step
- 02h Nonmaskable Interrupt (NMI)
- 03h Breakpoint Instruction
- 04h Overflow Instruction
- 05h Bounds Exception (80286, 80386)
- 06h Invalid Op Code (80286, 80386)
- 07h Math Coprocessor Not Present

PC Memory Map

FFFFF-F0000	ROM BIOS
EFFFF-E0000	ROM BIOS or EMS Window
DFFFF-D0000	BIOS Extensions, or EMS Window
CFFFF-C0000	BIOS Extensions or EMS Window
BFFFF-B0000	Video Memory (Text)
AFFFF-A0000	Video Memory (Graphics)
9FFFF-10000	Working RAM
0FFFF-00000	Working RAM, plus system information (eg. interrupt vector table)

Timer Channel

Channel	Used by
0	Internal system clock
1	Memory refresh
2	Speaker frequency

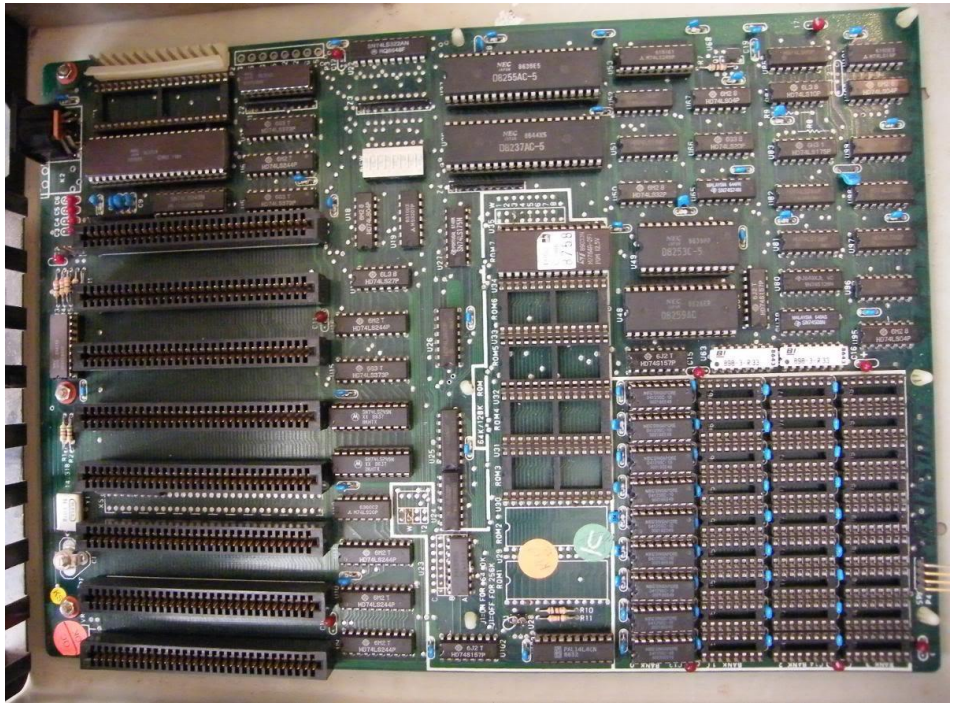
DMA Channel

Channel	Used by
0	Memory refresh
1	unused
2	Floppy disk controller
3	reserved
4	Cascade DMA1 -> DMA2
5	unused
6	unused
7	unused

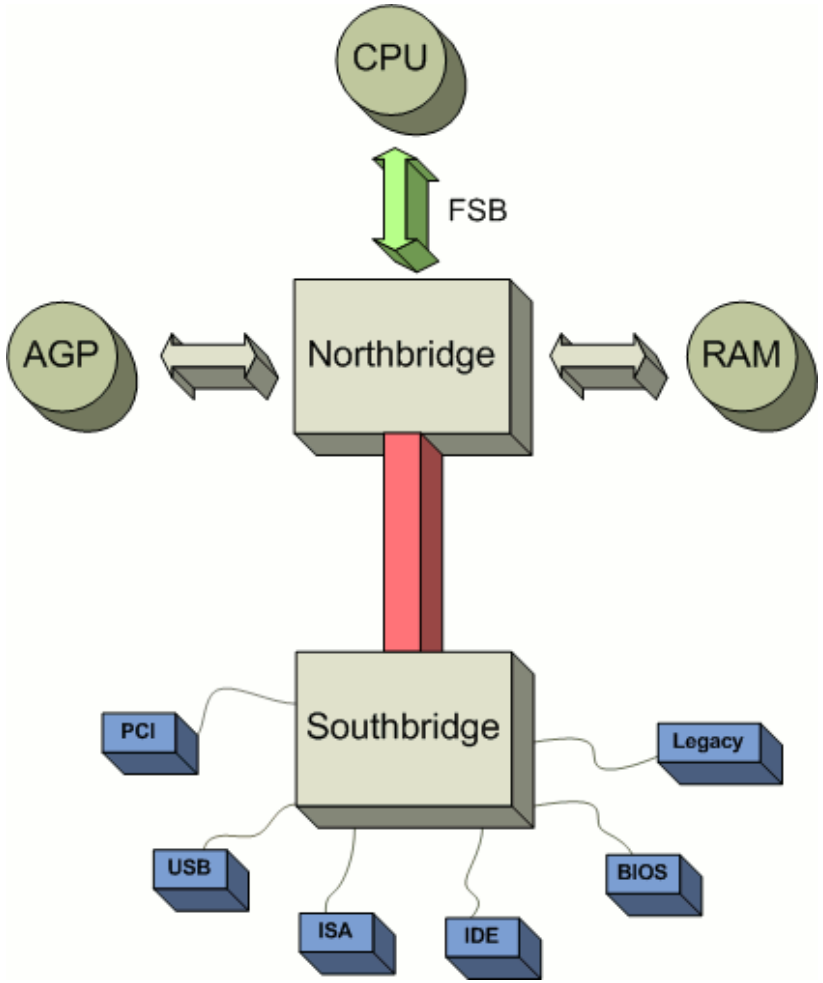
Note: Channels 5 –7 are 16 bit

Typical system based on 8088

- 8088 CPU
- clock generator/ controller
- DMA controller
- interrupt controller
- UART for communication
- Keyboard controller
- 8-bit wide memory
- LSI/MSI blocks: Decoders, transceivers, multiplexers



PC Chipset


CPU
FSB
AGP
Northbridge
RAM
Southbridge
PCI
Legacy
USB
BIOS
ISA
IDE

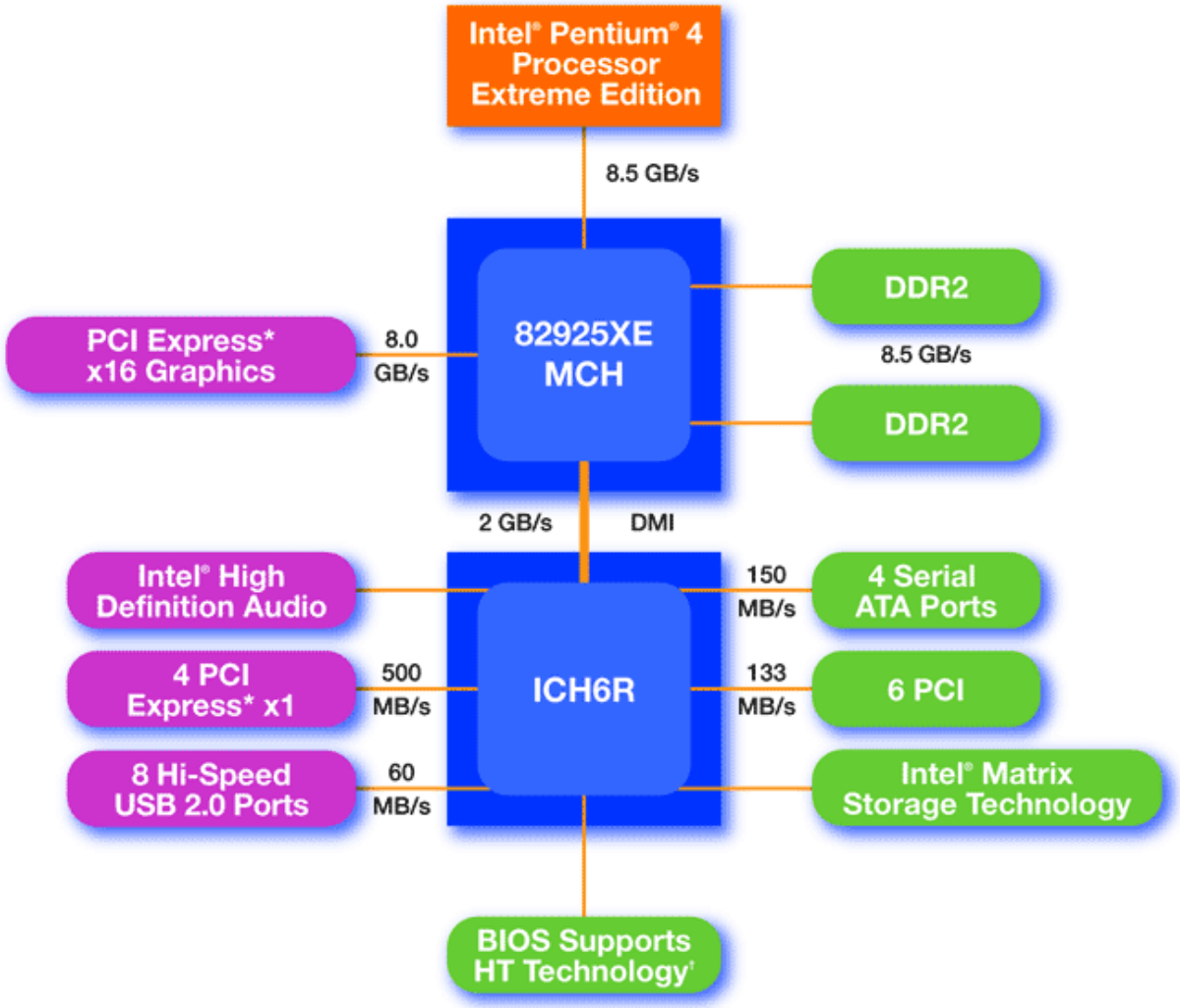
Northbridge

- Typically handles communications among
  - the CPU, RAM, BIOS ROM, and
  - PCI Express (or AGP) video cards,
  - and the southbridge.
- Different processors and RAM require different signaling,
  - a northbridge will typically work with
    - only one or two classes of CPUs and
    - generally only one type of RAM.
  - For example, i875 chipset will only work with
    - Pentium 4 or Celeron processors with clock > 1.3 GHZ
    - DDR SDRAM

Southbridge

- Southbridge contains controller integrated channel circuitry
  - It directly links signals from the I/O units to the CPU for data control and access via Northbridge
- Functionality Includes
  - PCI and ISA
  - SPI and SMBus
  - DMA, Interrupt controller and RTC
  - Power management and battery backed memory

Intel Chip Set



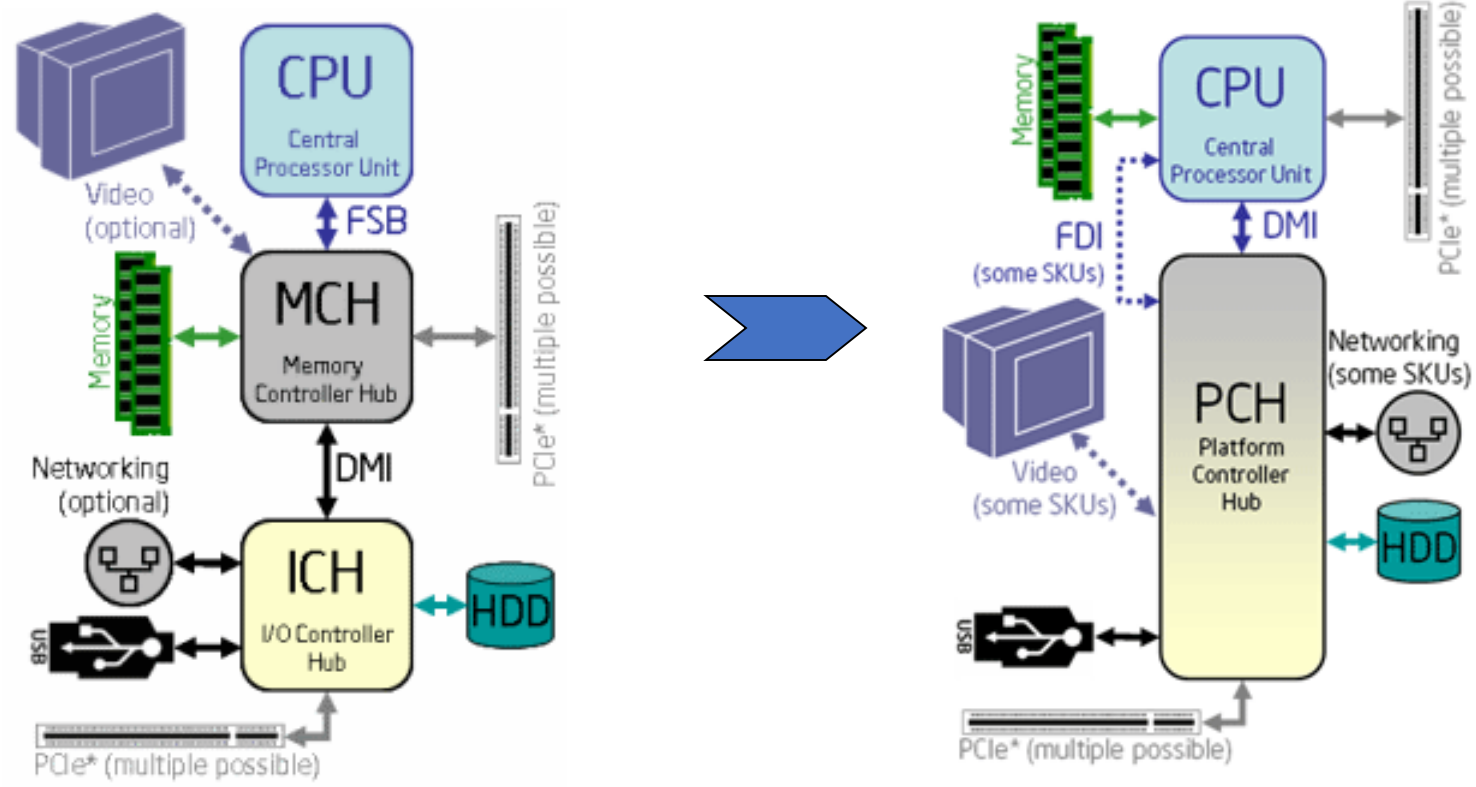
Recent Trends

- Basic trend in processor design has been to integrate more functions onto the chip, In order to
  - decreases overall motherboard cost
  - improves performance.
- The memory controller was moved onto the processor die
  - by AMD beginning with their AMD64 processors
  - by Intel with their Nehalem processors.
- Advantages is to reduce latency from the CPU-to-Memory so the CPU can control the memory directly

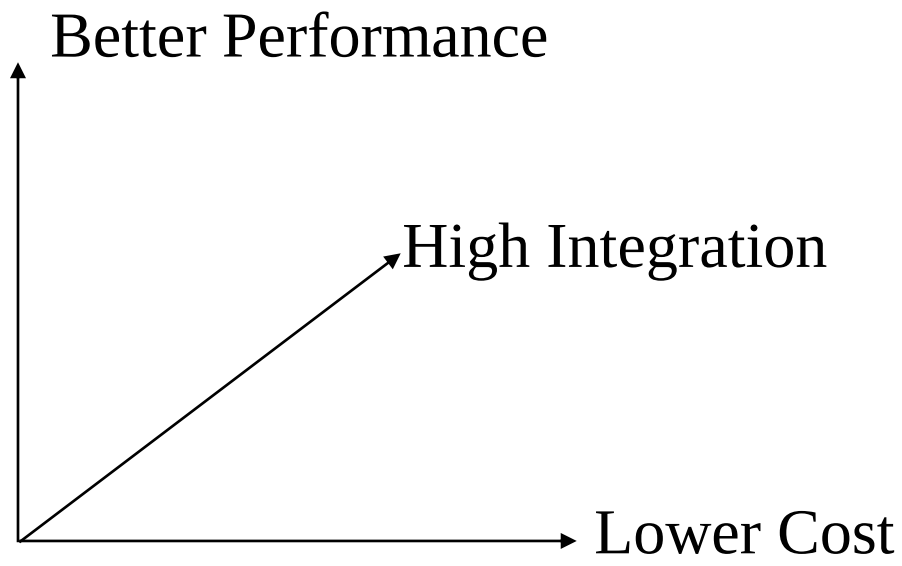
Recent Trends

- Nvidia's nForce3 chipset for AMD64 systems that is a single chip.
  - It combines all of the features of a normal southbridge with an AGP port and connects directly to the CPU.
  - On nForce4 boards it is called MCP (Media Communications Processor).
- Intel's "Sandy Bridge" processors feature full integration of northbridge functions onto the CPU chip
  - processor cores,
  - memory controller and
  - graphics processing unit (GPU)

Transformation



Three directions



Contributing factors

- Advances in VLSI: (Moore's law)
  - CPU and system controllers, other logic chips
- Memory
  - More memory at the same price
- Peripherals and I/O bus
  - More bandwidth at less cost

Advances in VLSI

- Process enhancements
  - Era of submicron feature size
    (0.09 micro in Pentium 4, Power4, Xilinx spartan3)
  - Interconnects

- How does it help?
  - Reduced die size improves frequency of operation, improves %yield
  - Requires reduced core voltage, results in lower power dissipation

PC Processors

PC processors - Earlier Generation

- 8086/ 8088 (CPU) + 8087 (FPU) + 8089 (IOP)
- 80286 + 80287
- 80386 + 80387
- 80486
  - 486SX, 486DX, 486DX2,486DX4

PC processors - New generation...

- Pentium: various speeds, with/ without MMX
- Pentium Pro
- Pentium-II
- Pentium-III
- Pentium-4
- 64 bit Core Microarchitecture (core 2)
- 64 bit Nehelam Microarchitecture (i3, i5, i7)
- Sandybridge, Haswell, Broadwell,Skylake (6^{th} generation)
- Itanium and Itanium-II

CPU

- What decides the CPU performance ?
  - ALU width (data size processed every clock)
  - clock speed (MHz)
  - Efficiency of the core (Number of instructions executed per second)
  - System interface (how well CPU interacts with the rest of the system)
- PC Processors employ a series of micro-architectural features to achieve extraordinary levels of parallelism and speed
  - enabled by astronomical transistor budgets

PC Processor features

- PC processors are
  - Super-scalar,
  - deeply pipelined,
  - out of order, and
  - speculative instruction execution
- most challenging performance obstacles
  - memory latency
  - branches

Path to Performance

- Task assigned to chip architects is to design
  - Highest-performance processor possible within a set of
    - cost
    - power
    - size constraints
- established by market requirements.
- Application performance is usually the best measure of success,
  - the market often mistakes clock frequency for performance

Path to Performance

- Making operations faster or executing
  - Advanced semiconductor process makes
    - Transistors switch faster
    - Signal travel faster
  - More transistors reduce execution latency
    - Eg. Full multiplier array vs. partial
- More of them in parallel

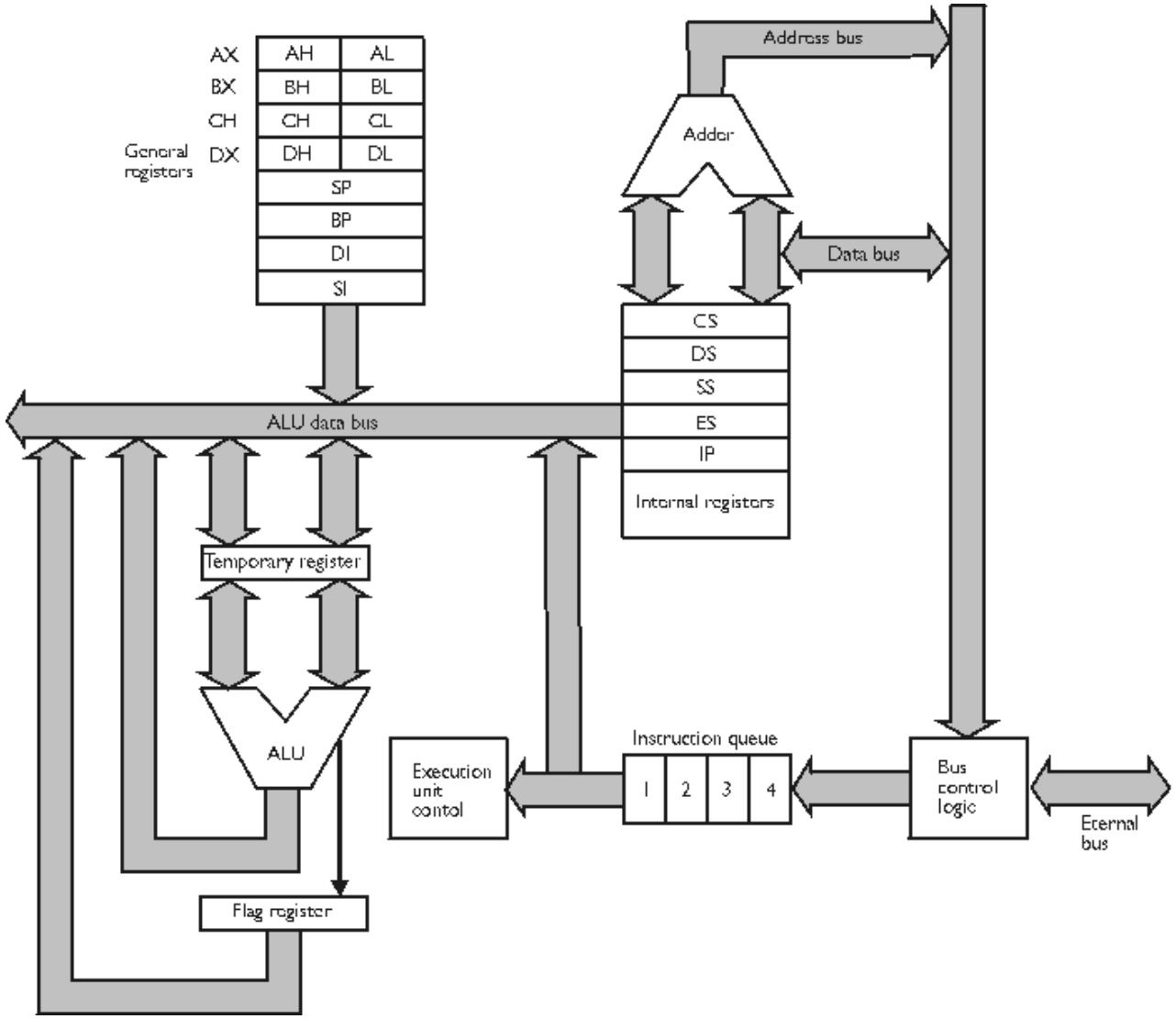
Intel: x86 architecture

- >25 years old! Origins in 8080 chip
- segmented memory approach to retain 8080 compatibility
- CISC architecture characterized by:
  - large no. of instructions
  - variable length instructions
  - large no. of addressing modes

8086/88

- 8086 in1976, 8088 in 1978: 29,000 transistors
- 16 bit CPU
- Memory addressability: 1 MB (20 addr lines),
- Data lines: 8/16
- ALU registers, segment registers, stack pointer registers, program counter
  - AX, BX, CX, DX
  - CS, DS, SS, ES
  - SI, DI
  - SP, BP, PC
- Frequency 4.77 MHz

8088 Internal Data Path



80286: multi-user configuration

- 1982, 134,000 transistors
- 16 bit CPU, faster core
- 24 bit address (16 MB), 16 bit data
- 8MHz .. 16 MHz clock speed
- real mode, protected mode
  - real mode: fast 8086
  - protected mode: multitasking, memory protection, virtual memory

80386: 32-bit computing

- 1985, 275,000 transistors
- 32-bit CPU
  - extended registers (eax, ebx,..)
- 32-bit addr bus, 32-bit data bus
- up to 33 MHz
- MMU: paging capability
- switch to/from protected mode
- Virtual x86 mode

80486: Functional Integration

- 1989, 1.2 Million transistors
- Vastly improved core, many of the instructions executed in one clock cycle
- On-chip FPU + 8 KB/ 8 KB instruction/ data cache
- Simplified and efficient bus interface
  - burst read/write (single address, multiple data)
  - Write buffer, write merge
  - support for external cache

486 variants

- 486 DX: 33, 40, 50 MHz
- 486 SX: low cost systems
- DX2 @ 66 MHz, DX4 @ 100 MHz
  - CPU clock runs at multiple of 33 MHz
  - external bus continues to run at 33 MHz
  - no motherboard re-design, low cost

Pentium

- 1993, 3.1 Million transistors, 32-bit CPU
- 32-bit external address, 64-bit external data bus
- ‘Superscalar’ implementation: up to two instructions per clock.
- Branch prediction
- 66 MHz (basic), 90, 120, 133+ MHz speed grades
- External bus at 66 MHz

Processor Design

- Microprocessors are Instruction Set Processors(ISP)
  - Executes instructions from a predefined instruction set
  - Functionality is characterized by the instruction set it is capable of executing
  - All programs that run on it are encoded in that instruction set
  - This predefined instruction set is also called as Instruction Set Architecture (ISA)

Instruction Set Architecture

- ISA serves as an interface between
  - Hardware and software
  - Programs and processor
- In terms of processor design methodology
  - ISA is the specification of a design
  - While a microprocessor or ISP is the implementation
- As is with any engineering design, ISP design is
  - Inherently a creative process
  - Involves subtle trade-offs
  - Requires good intuition and clever insights

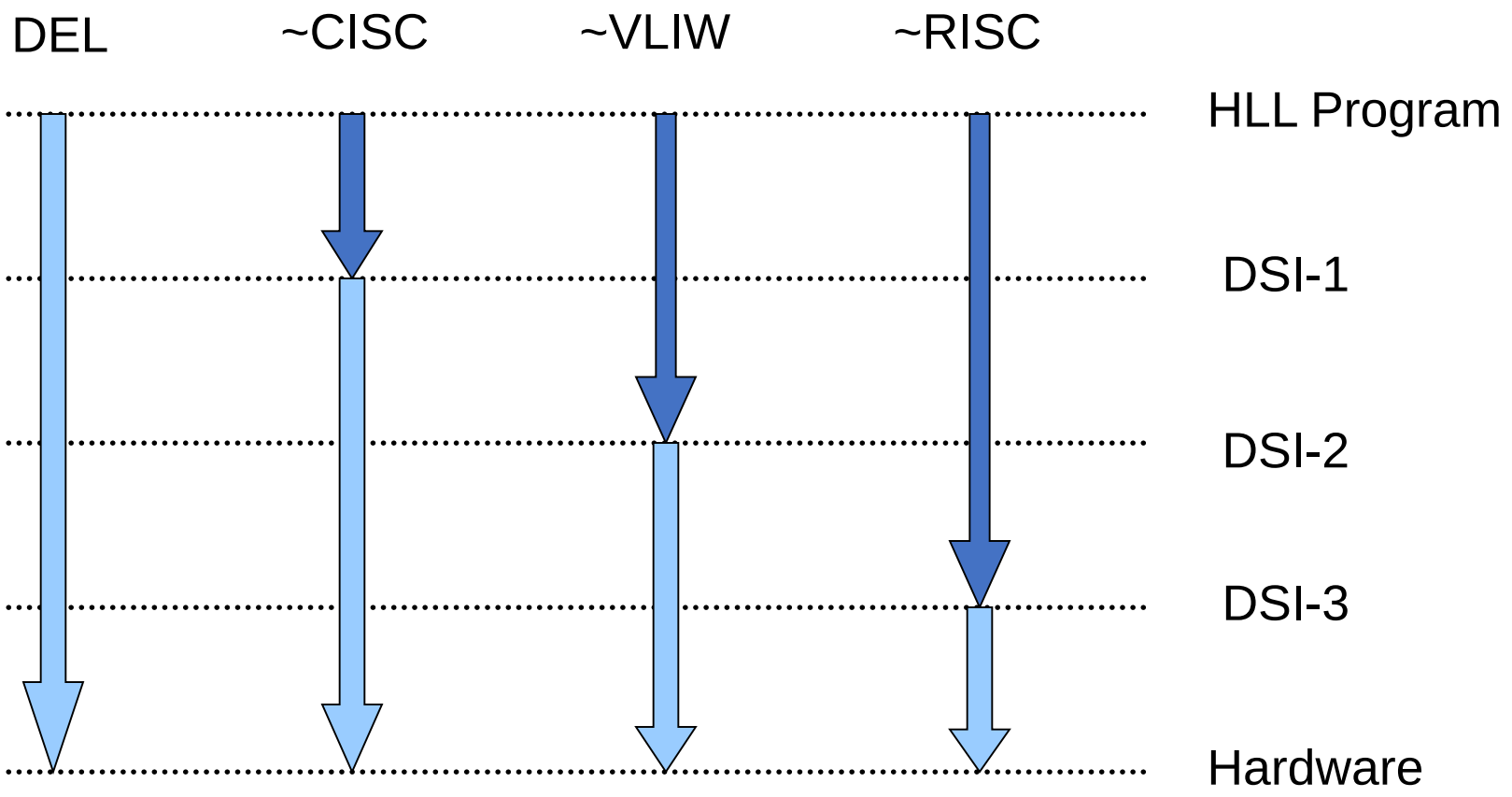
Instruction Set Architecture (contd.)

- Treating ISA as a interface
  - Programs can be developed without knowledge of actual implementation
  - Processors can be designed without concern for what programs will run on them
- Having ISA also ensures software portability
  - Program developed once will work during the lifetime of ISA
  - Migration to new ISA is very difficult
- Examples: IBM 360/370, Intel's IA32

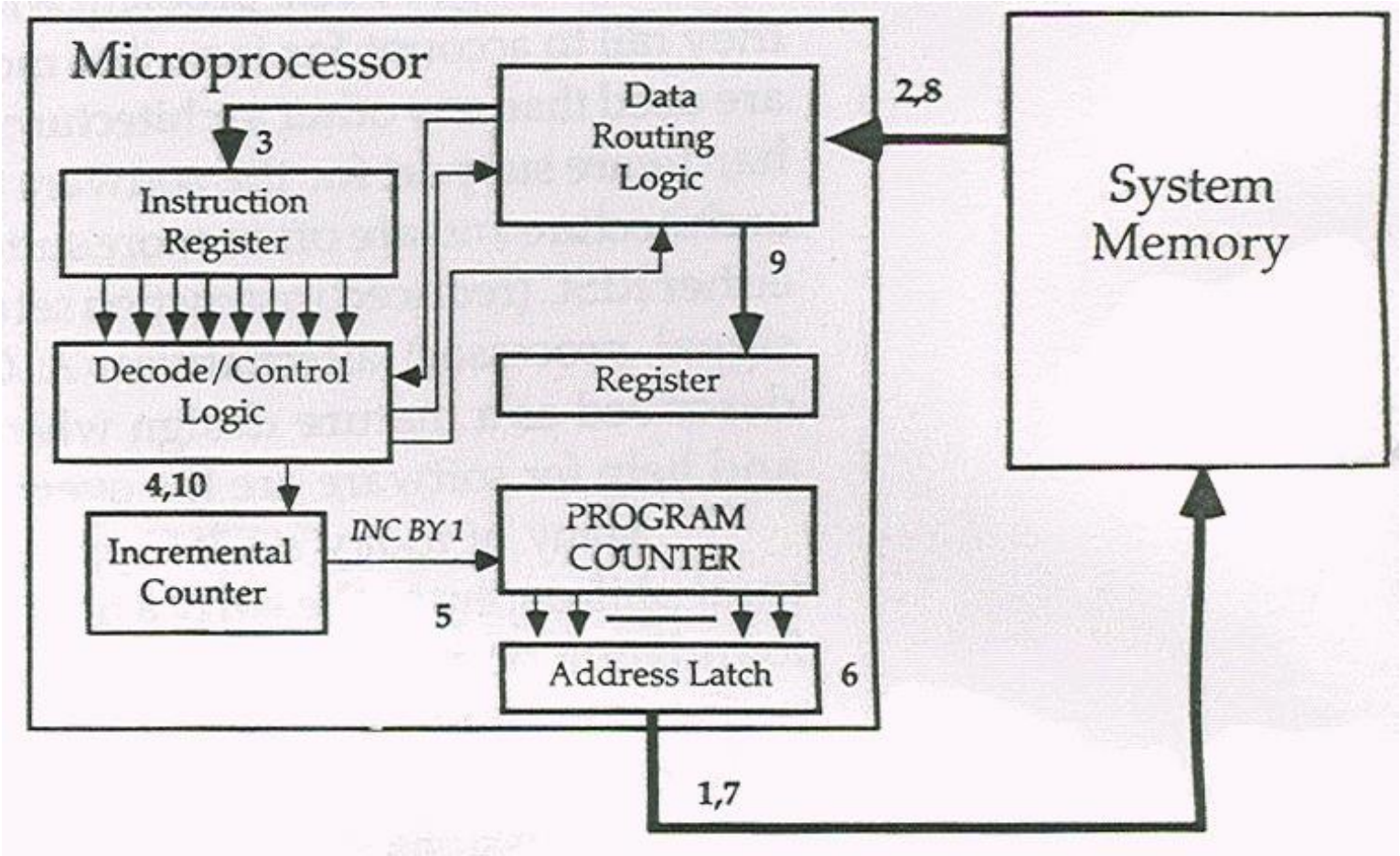
Dynamic Static Interface

- ISA as an interface separates
  - What is done statically at compile time
  - What is done dynamically at run time
- A key issue in design of ISA is placement of DSI in between
  - An application program written in High level language
  - Actual hardware of the machine
- There are different levels of abstraction, where the DSI can be placed

Possible Placements of DSI



Instruction Execution



Features of CISC

- Instructions for every possible eventuality
- Each instruction can indicate several low-level operations,
  - a load from memory, an arithmetic operation, and a memory store
- Variable instruction length
- Widely differing instruction execution times
- Realisation is complex
- ALU is microcoded
- Examples: Intel x86, Motorola 68000

RISC

- 80% of ‘typical’ programs use 20% of instruction set
- Majority of "orthogonal" addressing modes are ignored by most programs.
- Most used instructions are speed optimized, others are removed
- CPU design is simplified
  - Superscaler implementation becomes easy
- CPU is hardwired(not micro-coded)

RISC

- All instructions are executed in one cycle, represented by one word
  - Use of pipeling
- Memory is accessed by load/store architecture
- Silicon saved is used for FPU and or Cache
- Examples ARM, PA-RISC, SPARC, MIPS, PowerPC.

Performance Equation

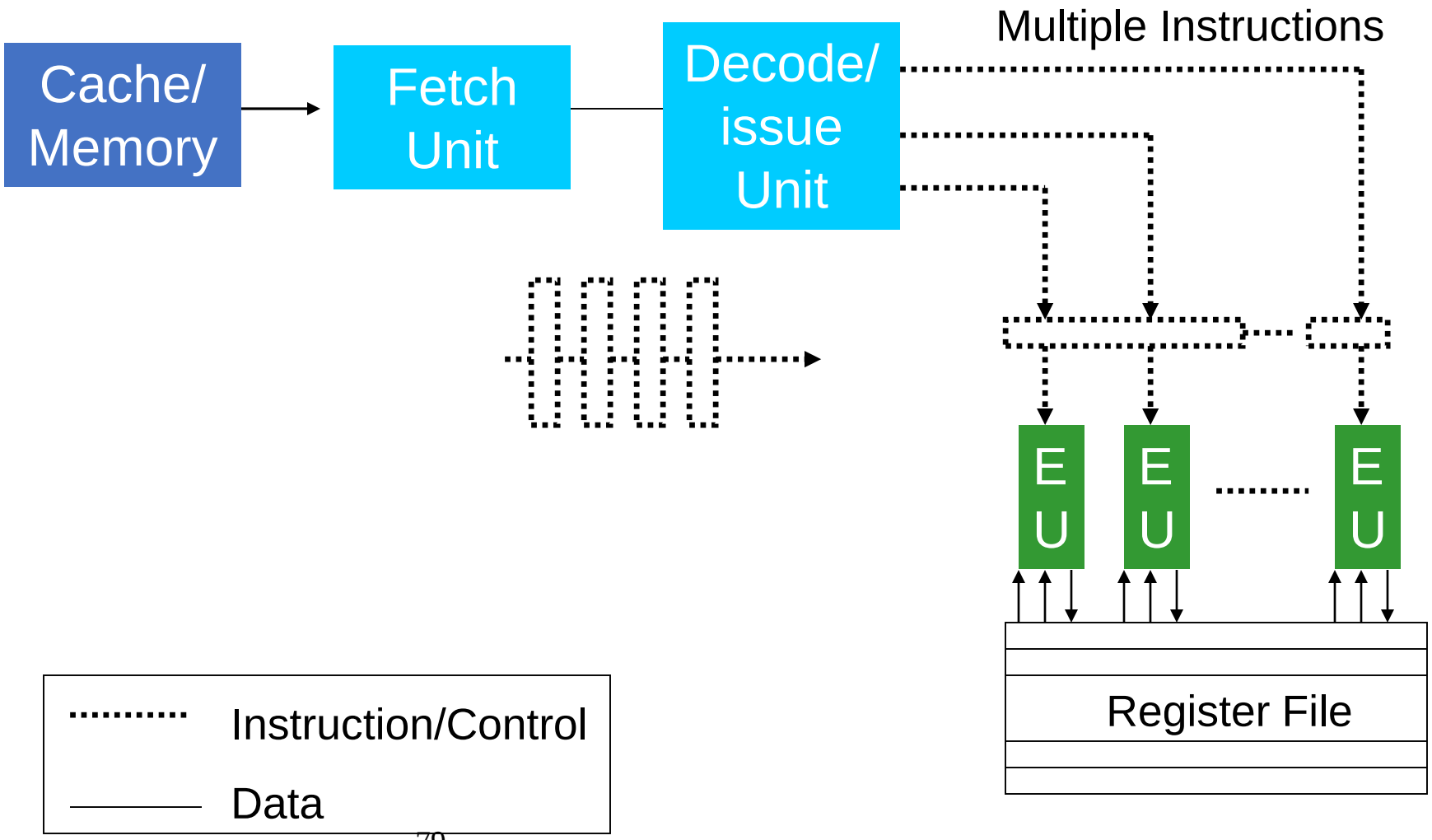
$$\frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{instructions}}{\text{program}}$$

- CISC
  - minimises instruction/program
  - Sacrifices cycles/instruction
- RISC (does the opposite)
  - minimises cycles/instruction
  - Resulting in increase in instruction/program

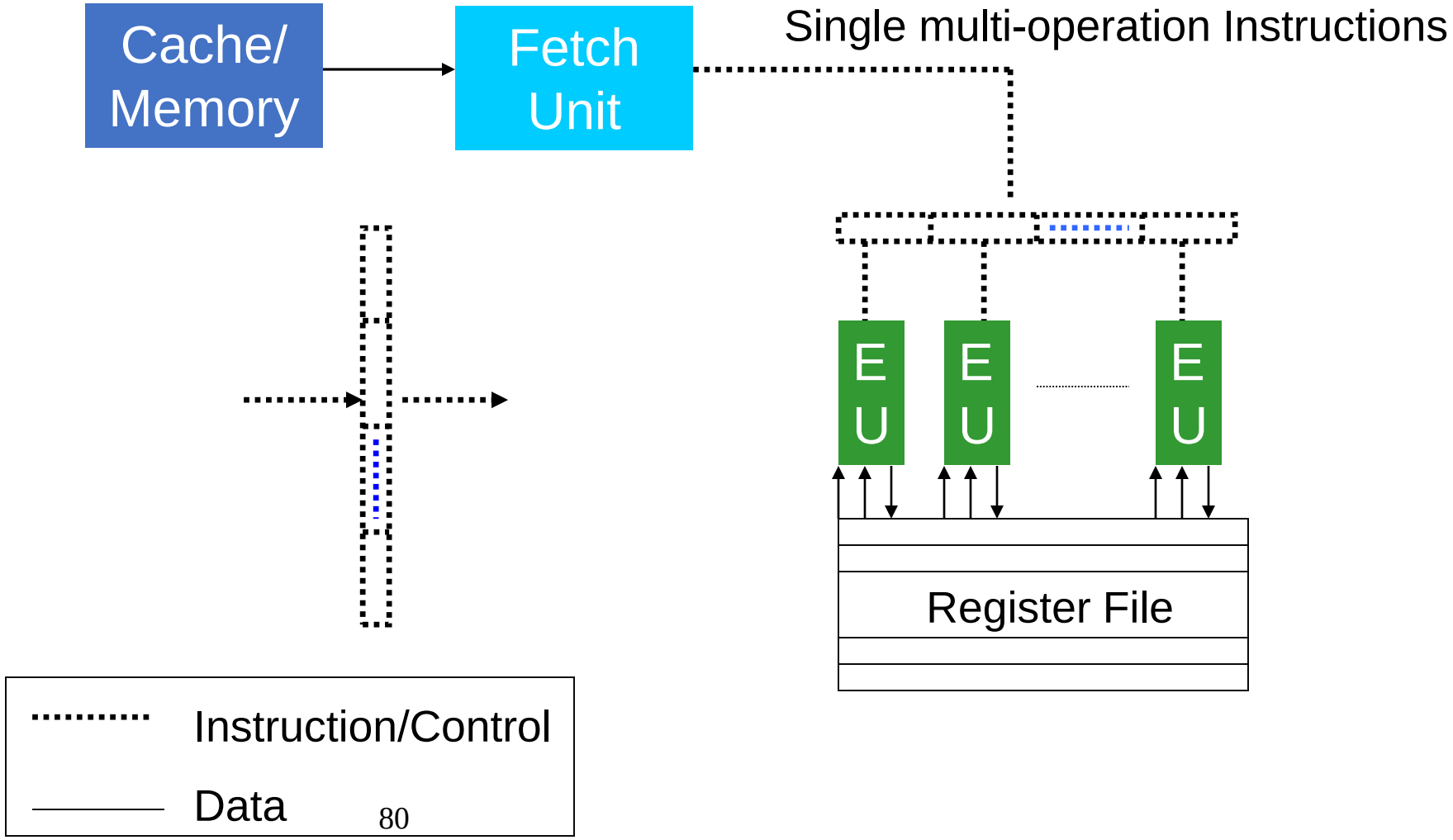
ILP Processors

- Instruction Level Parallel Processors
  - Pipelined
    - A new instruction enters every clock
    - Instruction parallelism = No. of pipeline stages
  - Superscaler
    - Multiple execution units
    - Sequential instructions, multiple issue
  - VLIW
    - Multiple execution units
    - Multiple instructions packed in one word

Superscaler approach



VLIW approach



History of SuperScalar

- Prototypes
  - Cheetah (’82-83)and America(’85-) projects by IBM
  - Multititan(’85-87) by DEC
- Commercial
  - i960CA embedded RISC processor by Intel in 1989
  - MC88000, HP-PA, Sun SPARC, AM29000
- Superscalar was introduced earlier in RISC than CISC

CISC superscalar

- Making CISC superscalar has higher complexity
  - Decoding multiple variable length instructions
  - Implementing memory architecture
- Pentium and MC 68060, first implementations
- Result of converting existing lines into superscalar
- K5 was designed from scratch as x86 compatible
- Low issue rate of 2 instructions per cycle
- Pentium Pro has superscalar RISC core
  - Issue rate of 4, equivalent of CISC rate of ‘about 2’

SuperScalar Processing

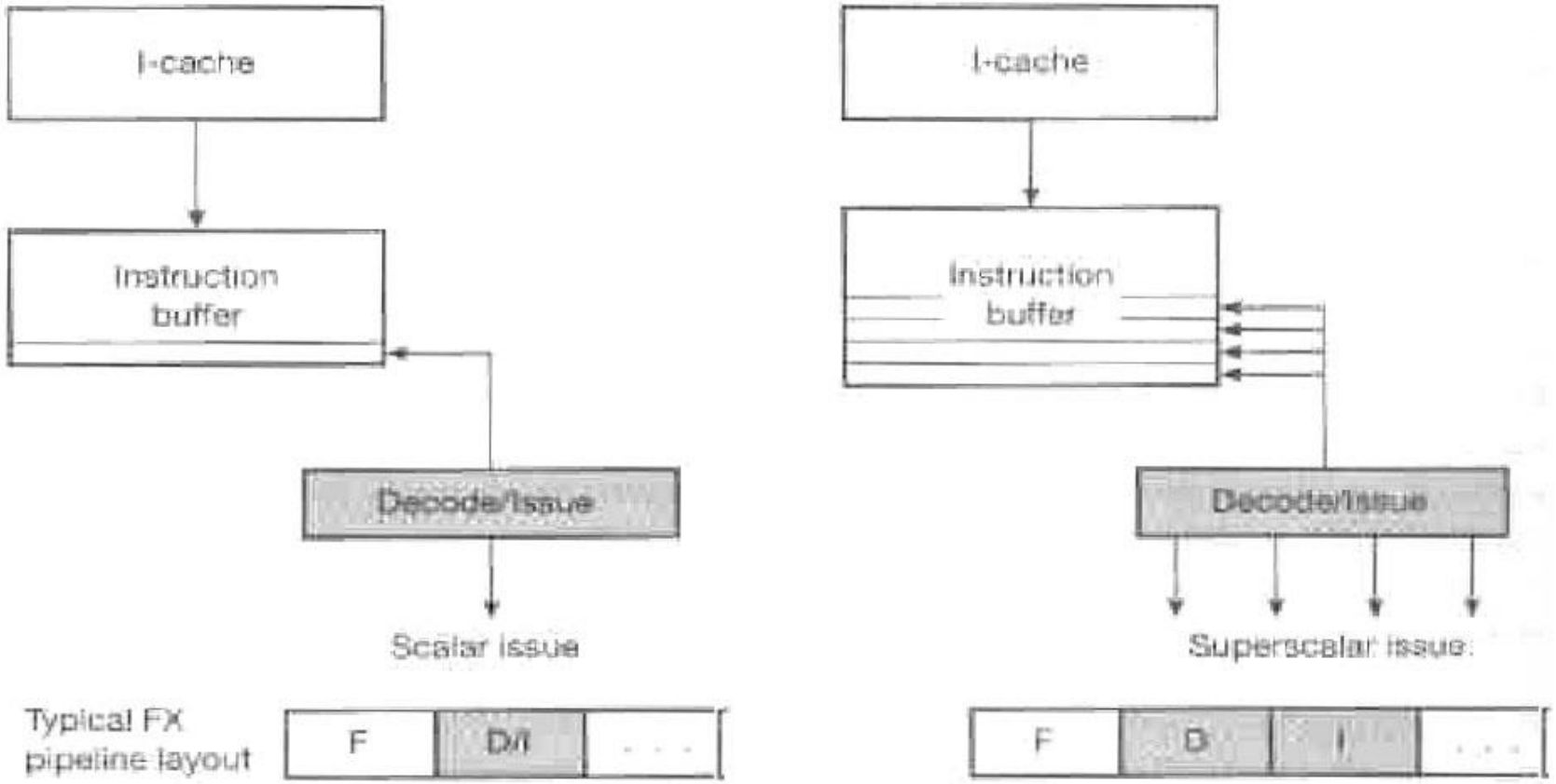
- Five specific tasks
  - Parallel decoding
  - Superscalar instruction issue
  - Parallel instruction execution
  - Preserving the sequential consistency of execution
  - Preserving the sequential consistency of exception processing

Parallel Decoding

Parallel Decoding

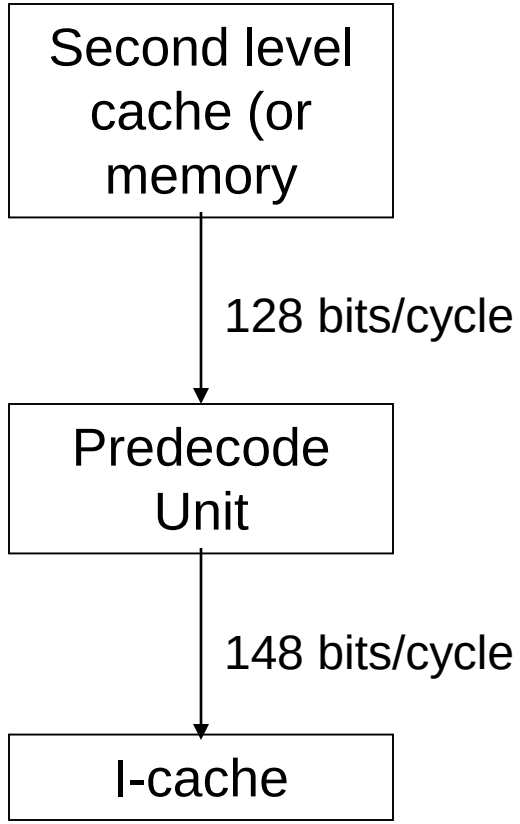
- To issue multiple instructions per cycle, first task necessarily is parallel decoding
- More complex than scalar, increases with issue rate
- Needs to check the dependencies in pipelined processor to decide instruction can be issued
  - Compare with instructions currently in execution
  - Instruction candidates for next issue
- More EUs result in more comparisons for checks
- To achieve higher clock frequency, the decode-issue path tend to use 2 to 4 pipeline cycles (stages)

Decoding and Issue



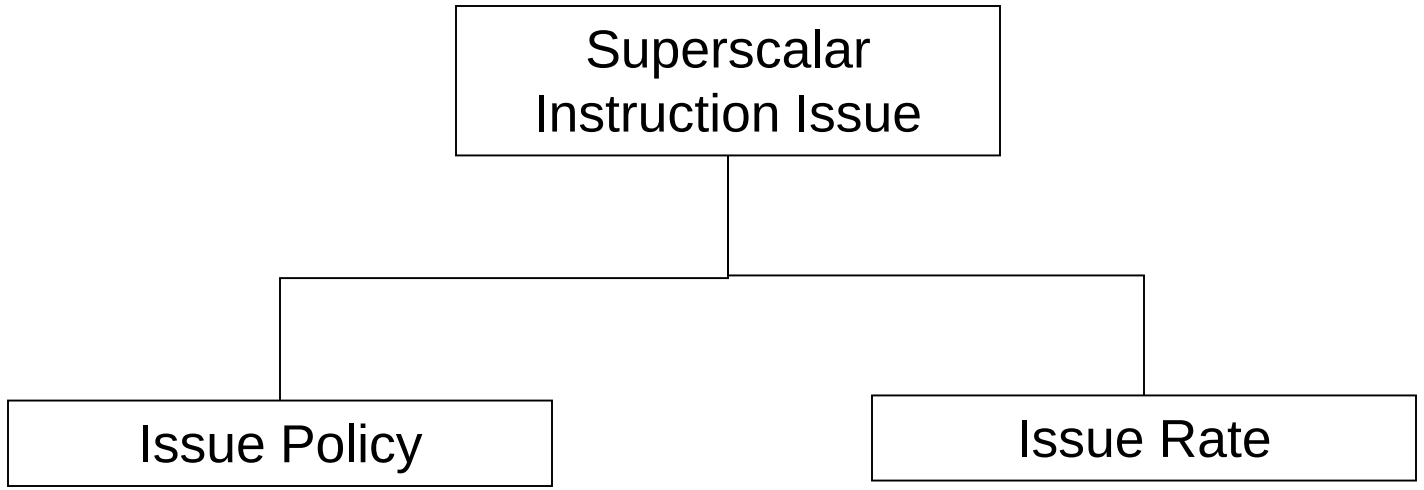
Predecoding

- Shifts a part of the decode task into loading phase of on-chip instruction cache (i-cache)
- Predecode unit performs partial decoding to append 4-7 decode bits to each instruction indicating
  - Instruction class
  - Type of resources required for execution

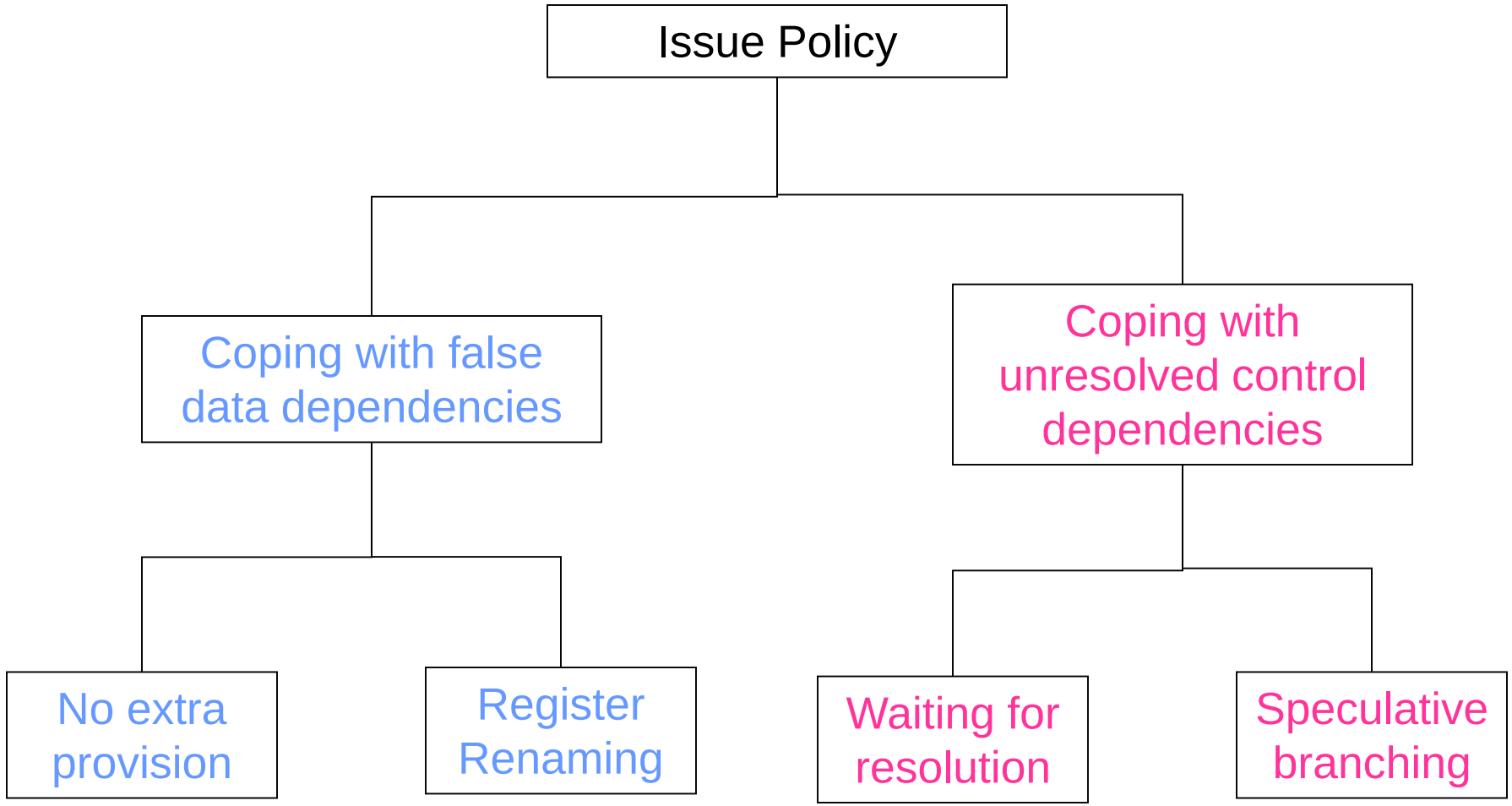


Superscaler Instruction Issue

Instruction Issue



Issue Policy



Data Dependecies

- If two sequential instructions have a common operand, they are data dependent
  - Except when the common operand is a source
- Eg. Next instruction uses previous result as a source
- True Dependency
  - Read After Write (RAW)

```
i1: load r1, a
i2: add r2, r1, r1
```

  - I2 cannot be executed correctly until i1 loads r1

False Data dependencies

- Write After Read (WAR)

```
i1: mul r1, r2, r3
i2: add r2, r4, r5
```

- i2 writes r2 while i1 uses r2 as a source
- For any reasons, i2 is executed before i1, then r2 would be re-written earlier than it is by i1
- Write After Write (WAW)

```
i1: mul r1, r2, r3
i2: add r1, r4, r5
```

- I1 and i2 both writes r2

Register Renaming

- If WAR or WAW is encountered, the destination register causing the dependency is renamed
  - Result is written in the dynamically allocated ‘spare register’
- Static Implementation
  - Performed during compilation
  - By parallel optimising compilers
- Dynamic implementation
  - Performed during execution
  - Requires extra logic for registers, data path and logic

Unresolved control dependencies

- Conditional control transfer instruction cause problem
  - Condition has not been produced
  - Condition needs to be evaluated
- Solution
  - Wait till the referenced condition becomes available
  - Employ speculative execution

Issue Rate

- Also known as degree of superscalarity
- Maximum number of instructions a processor can issue in same cycle
- A higher issue rate offers a higher performance potential
  - Implementation requires more complex logic

Parallel Execution

Parallel Execution

- Instructions executed in parallel will generally be finished in out of program order
- It does not matter if instructions are dispatched in-order or out-of-order
  - unequal execution times force instructions to finish out of order
- To finish: required operation is over, except for writing back the result into register/memory
- To complete: Writing back the result into register
- To retire: write back the result and delete the completed instruction from ROB entry

finish, complete, retire

Required operation is completed

Finish

- Result written in Architectural register or memory
- updated status bits

+

- Delete completed instruction from ROB

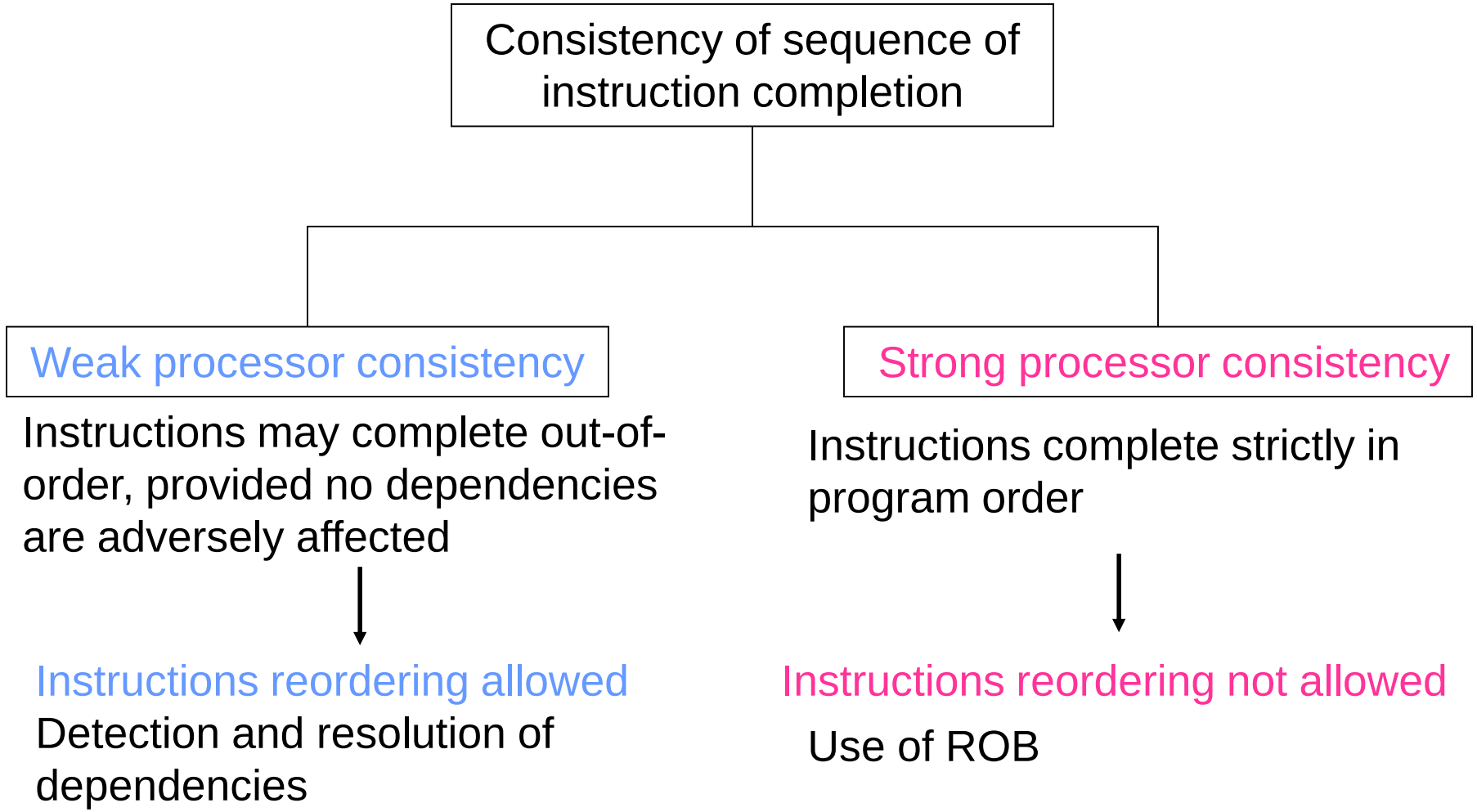
Complete

Retire

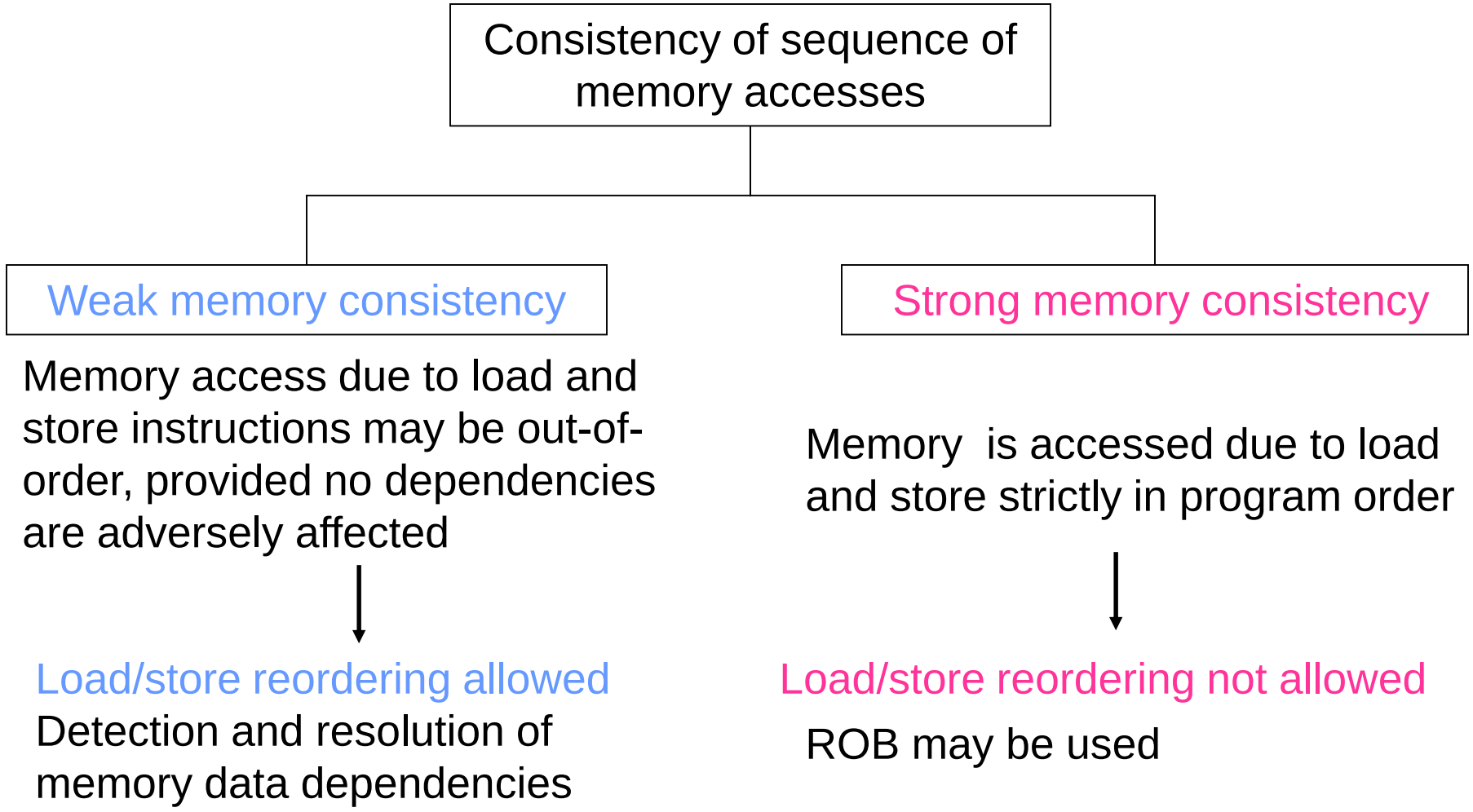
Sequential consistency models

- Instructions may finish in-order or out-of-order
  - Overall instruction execution must mimic sequential execution
- Sequential consistency relates to
  - Order in which instructions are completed (processor consistency)
  - Order in which memory is accessed (load/store) (memory consistency)

Processor Consistency



Memory Consistency



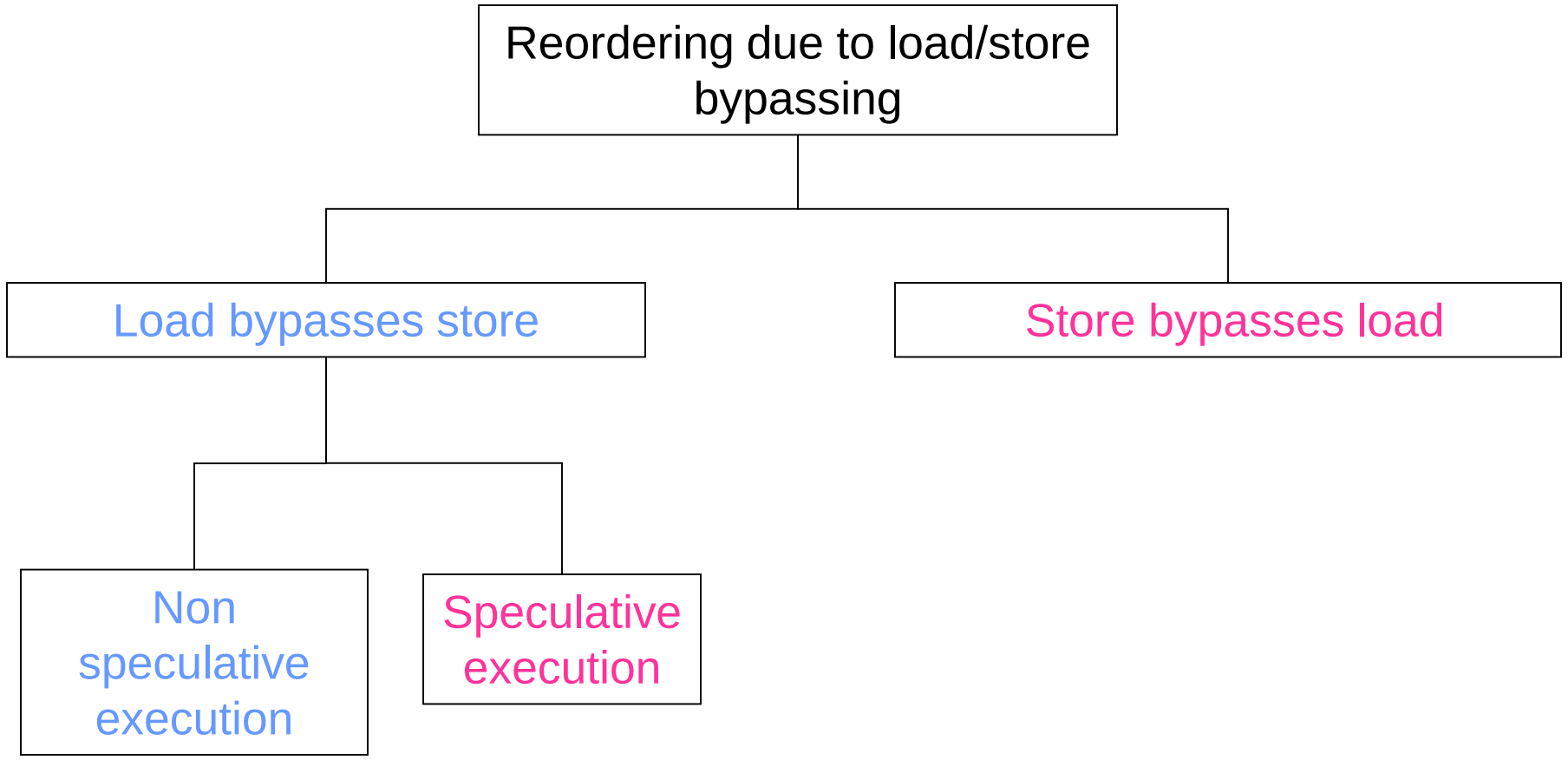
Load/Store reordering

- Actions of load/store affect both processor and memory
- Load execution
  - Address is computed
  - Get data from cache into buffer register (state:finish)
  - Write data into architectural register(complete)
- Store execution
  - Address is computed
  - Wait for operands to be available (state:finish)
  - Wait for turn in ROB, then forward address and data to cache (assuming ROB in use)

Load/Store reordering

- Weak memory consistency allows reordering of memory access. Advantages:
  - Permits load/store bypassing
  - Makes speculative loads feasible
  - Allows cache misses to be hidden
- Load/store bypassing
  - Either load can bypass pending stores or vice-versa, provided data dependencies are not violated
- Load bypassing store can be done either non-speculatively or speculatively

Memory access reordering



Load bypass store

- Allows runtime overlapping of tight loops
  - Loads are permitted at the beginning of iteration, without having to wait until stores at the end of previous iteration are completed
- To avoid fetching false data value, none of the preceding pending stores should have same address
- To fulfill this, address of load has to be compared with addresses of all pending stores
- Address of some pending stores may not be available
  - Delay permission for load (Non-Speculative)
  - Allow speculative loads

Speculative Loads

- Memory access started in spite of unresolved address checks
- For vast majority of bypassed loads the addresses of subsequently computed preceding stores do not match
  - Hence speculative behavior is justified
- If there is a match, speculative loads must be undone
- When store addresses have been computed, they are compared against that of all younger loads
- If hit is found, the load instruction and all subsequent instructions are cancelled and load is re-executed

ReOrder Buffer

- ROB was originally conceived for solving precise interrupt problem
  - Now uses as a tool for preserving sequential consistency for multiple EUs operating in parallel
- ROB is a circular buffer with two pointers
  - Head: where instructions are added
  - Tail: where instructions are retired
- Instructions are written in strict program order

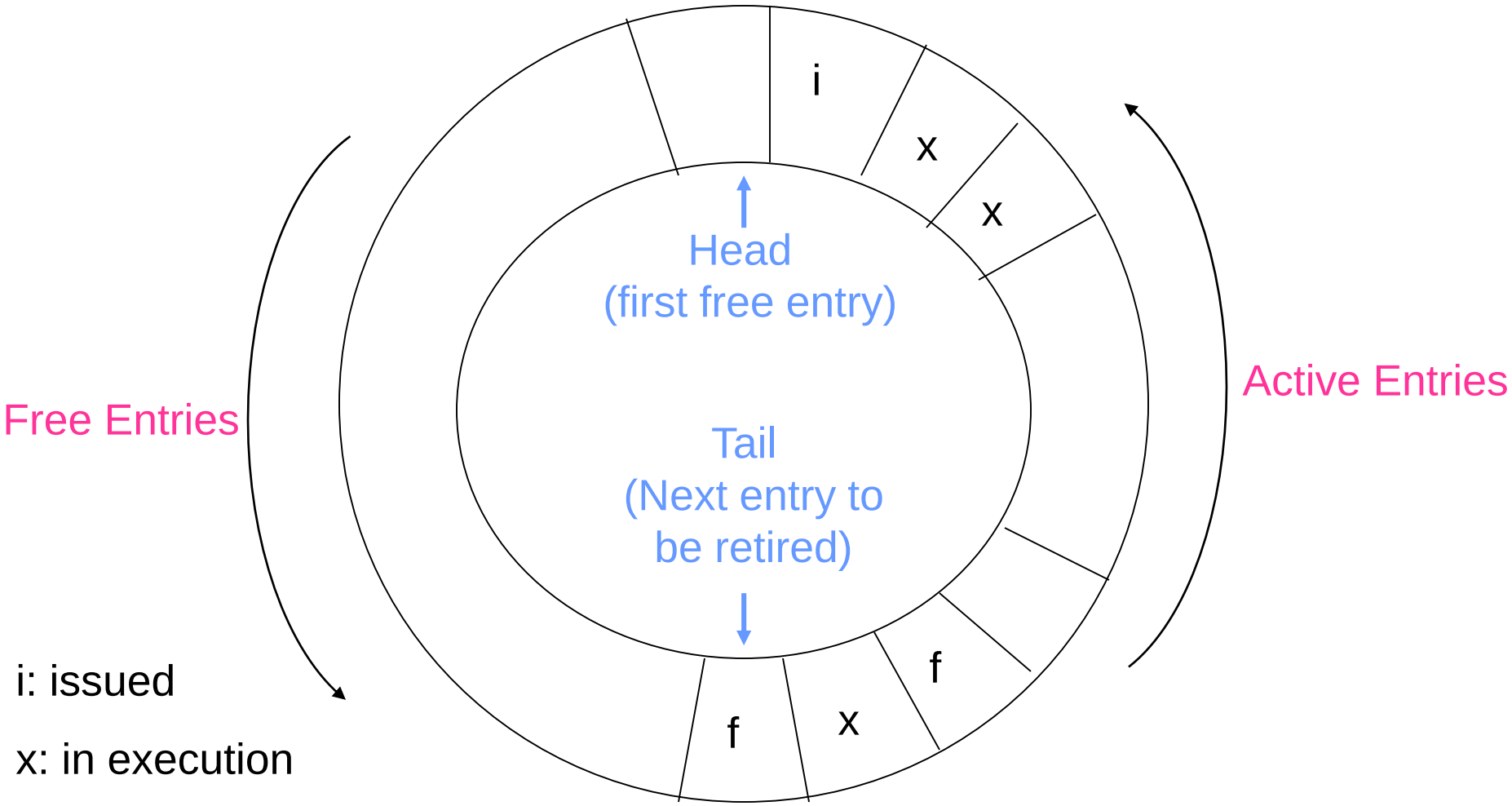
ROB

- Status indicates
  - I: instruction is issued
  - X: in execution
  - F: finished
- An instruction is allowed to retire only if
  - It has finished
  - All previous instructions are retired
- This ensures all instructions retire strictly in order
- Sequential consistency is maintained since only retiring instructions are allowed to complete

ROB

- ROB is useful for speculative execution
  - Each entry has additional status field
- Finished instructions are not allowed to retire
  - if they are in speculative state
- When condition is resolved
  - Speculative execution is found true or false
- If true, in-order completion is maintained
- If false, instructions are cancelled

ROB



i: issued

x: in execution

f: finished

Thank You

End of part 1

ashishk @ cdac.in