Lecture 2: Registers Instruction set

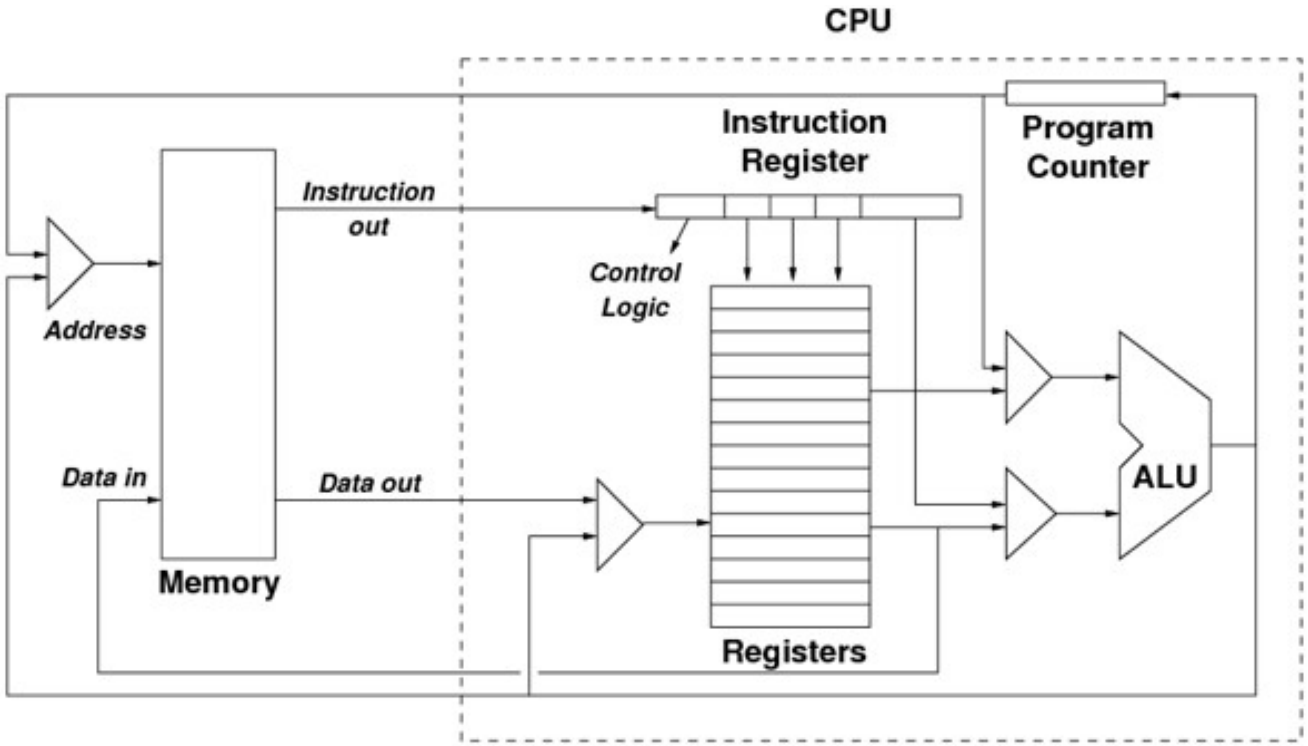
Computer Architecture

Roy Vellaisamy
Professor of Intelligent systems

https://www.gla.ac.uk/schools/engineering/staff/royvellaisamy/

Registers

- A register is a temporary storage area built into a CPU. Some registers are used internally and cannot be accessed outside the processor, while others are user-accessible. Most modern CPU architectures include both types of registers.
- Register size varies from 8 to 64bit in most products, and the number of registers depends on the CPU architecture.



Registers

- Some key terms (names) of register widely used:
- **Accumulator** – It stores the "value" "ready for" or "just after" process
- **Memory Address Register (MAR)** – Store the "location" in memory that the content is fetched (data or instructions)
- **Memory Buffer Register (MBR)** – The data or instructions that is sent to or received from the memory
- **Instruction Register** – The instruction of the work is stored. When the instruction is completed, the next instruction is fetched from the memory for processing
- **Program Counter (PC)** – Sometimes as "Instruction Pointer(IP)", it stores the address of the next instruction that will be executed. The processors usually fetch instructions sequentially, but it can be branched to some other addresses by placing a value.
- General Purpose Registers – It is to store data and intermediate results during program execution.

Shift Register

Shift register is a special register that can "shift" the content serially.

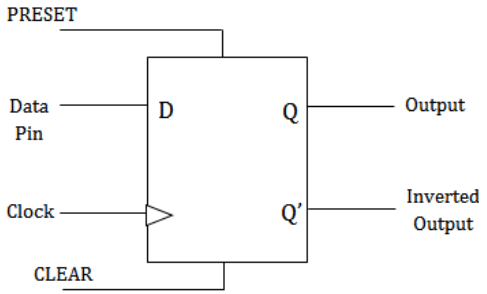
Type of Shift register:

- Serial In – Serial Out
  - Mainly work as delay line, or re-shape the waveform
- Serial In – Parallel Out
  - Convert serial data to Parallel data
- Parallel In – Serial Out
  - Convert parallel data to Serial data
- Parallel In – Parallel Out
  - Work as a delay line and waveform shaper, in most case it includes the bi-directional shift function as a x2 (left shift) or /2 (right shift)
- Ring counter
  - It is used to count data in a continuous loop, various stages connected together to provide a divider circuit.

D Flip-Flop

D Flip-Flop is to push the state of input "Data" to output "Q". Below is an example of edge trigger D Flip Flop. The state of Q is changed to same state as D when the "clk" at rising edge.

It is a major component of register.

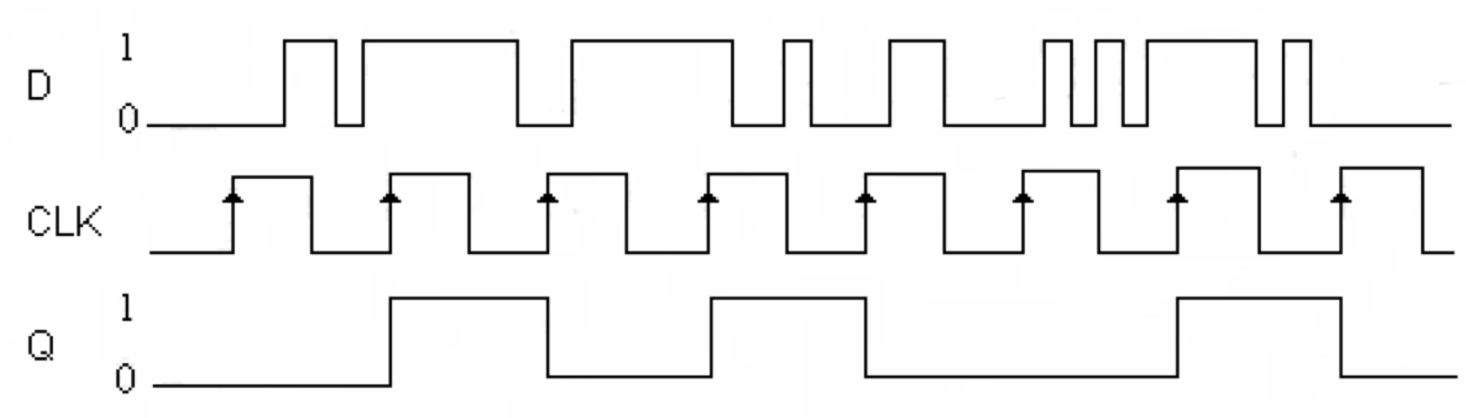


Symbol: D Flip-flop

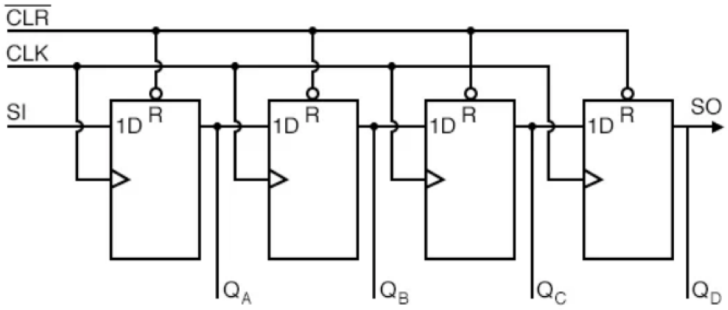
D Flip Flop Truth Table

CL (Note 1)	D	R	S	Q	$\overline{Q}$
⌐	0	0	0	0	1
⌐	1	0	0	1	0
¬	x	0	0	Q	$\overline{Q}$
x	x	1	0	0	1
x	x	0	1	1	0
x	x	1	1	1	1

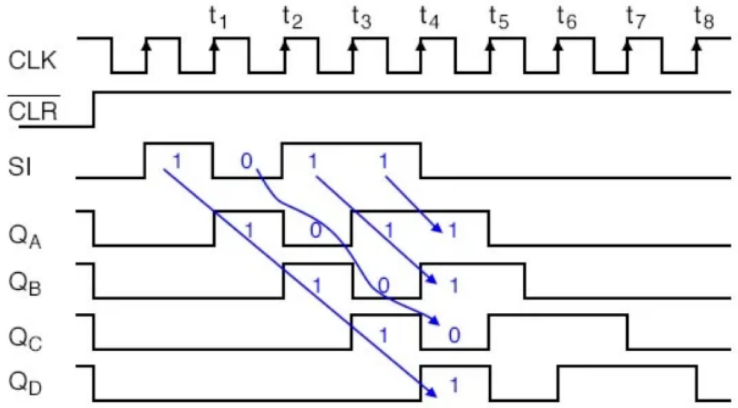
No Change
x = Don't Care Case
Note 1: Level Change



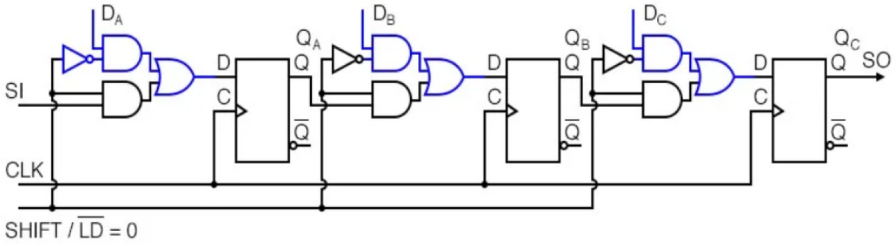
Serial In – Parallel Out & Parallel In – Serial Out



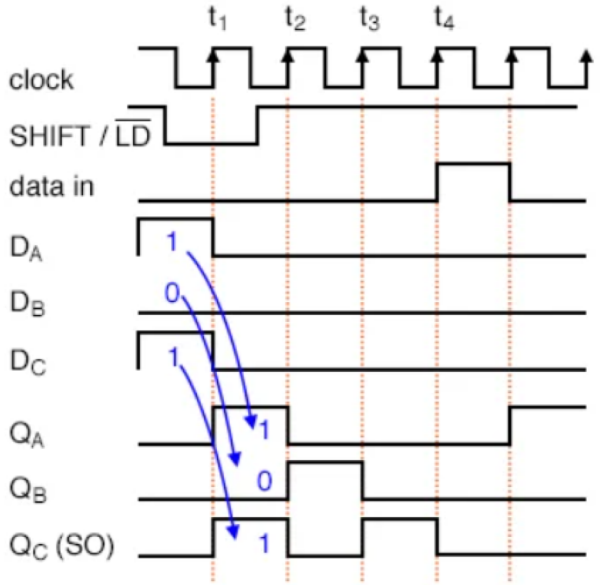
Serial-in/ Parallel-out shift register details



Serial-in/ parallel-out shift register waveforms

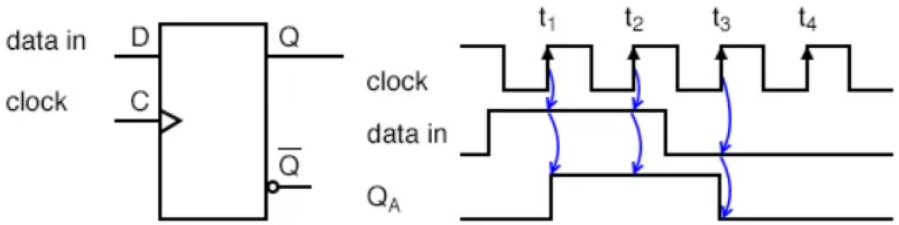


Parallel-in/ serial-out shift register showing parallel load path



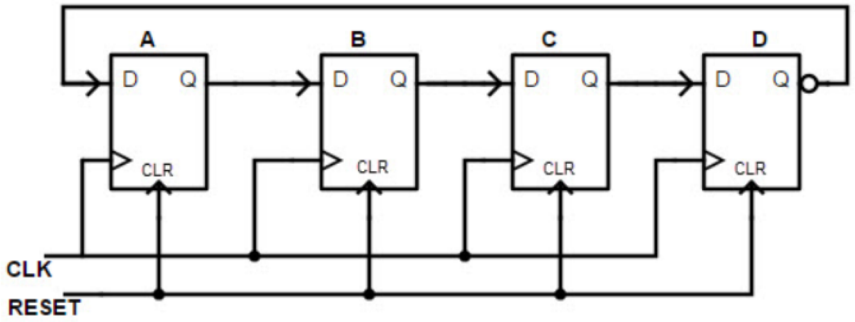
Parallel-in/ serial-out shift register load/ shift waveforms

Serial in – Serial out, Parallel in – Parallel out, Ring counter

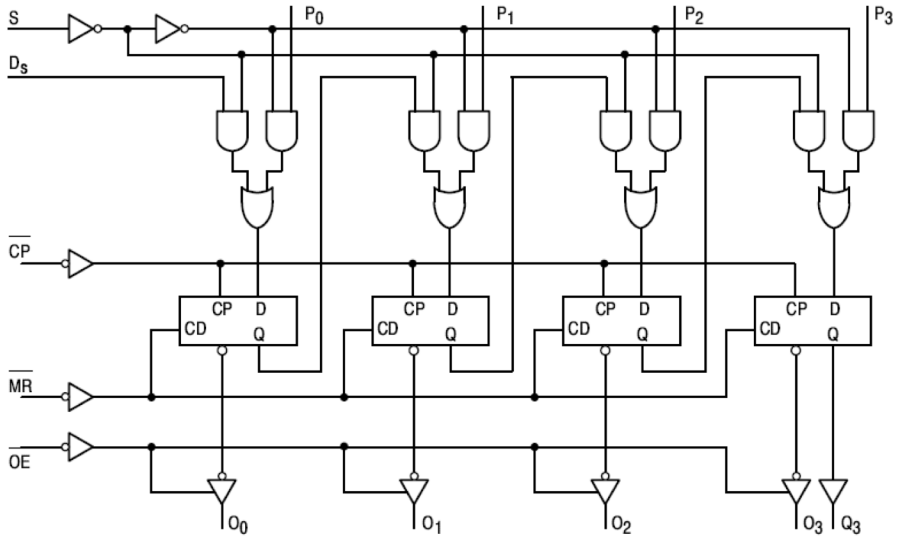


Data present at clock time is transferred from **D** to **Q**.

1 stage Serial in – Serial out



Ring Counter



74LS395

Parallel in – Parallel out Register

Registers summary

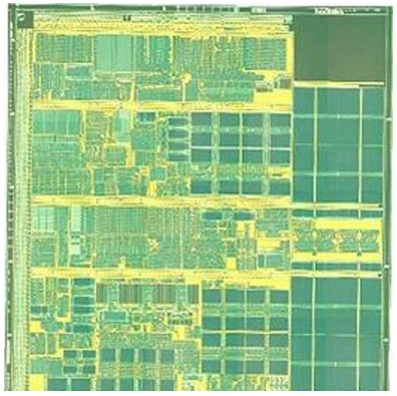
- A register is a special state machine that stores multiple bits of data.
- Several variations are possible:
  - Parallel loading to store data into the register.
  - Shifting the register contents either left or right.
  - Counters are considered a type of register too!
- One application of shift registers is converting between serial and parallel data.
- Most programs need more storage space than registers provide.
  - We'll introduce RAM to address this problem.
- Registers are a central part of modern processors.

Registers in Modern Hardware

- Registers store data in the CPU
  - Used to supply values to the ALU.
  - Used to store the results.
  - The speed can match the ALU and others core logic
- If we can use registers, why we need RAM?

CPU	GPR's	Size	L1 Cache	L2 Cache
Pentium 4	8	32 bits	8 KB	512 KB
Athlon XP	8	32 bits	64 KB	512 KB
Athlon 64	16	64 bits	64 KB	1024 KB
PowerPC 970 (G5)	32	64 bits	64 KB	512 KB
Itanium 2	128	64 bits	16 KB	256 KB
MIPS R14000	32	64 bits	32 KB	16 MB

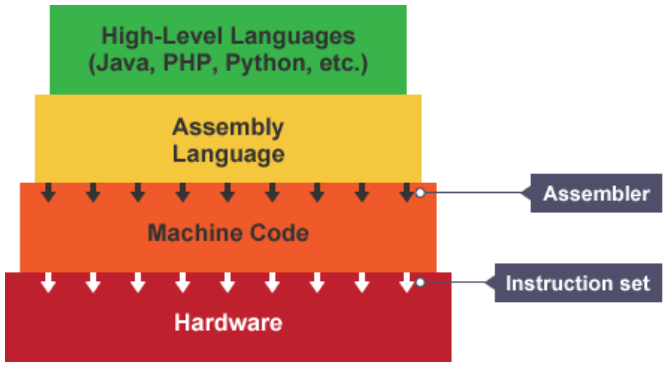
Answer: Registers' size and power consumption is much larger than RAM !



- Registers occupy the most expensive space on a chip – the core.
- L1 and L2 are very fast RAM – but not as fast as registers.

What is an Instruction Set?

- The complete collection of instructions that are understood by a CPU
- Machine Code
- Binary
- Usually represented by assembly codes



Operation	Opcode	Meaning of other bytes		
ADD	0x20	Dest. reg.	Reg. 1	Reg. 2
IMM	0x60	Dest. reg.	Unused	Value
BNE	0x11	Dest. address	Reg. 1	Reg. 2
LOAD	0xA4	Dest. reg.	Reg. with address	Unused
STORE	0x08	Unused	Reg. with dest. address	Source reg.

```
0x 60 00 00 80   IMM   R0,   0x80
0x A4 00 00 00   LOAD  R0,   R0
0x 60 01 00 84   IMM   R1,   0x84
0x A4 01 01 00   LOAD  R1,   R1
0x 60 02 00 00   IMM   R2,   0x0
0x 60 03 00 04   IMM   R3,   0x4
0x 60 04 00 00   IMM   R4,   0x0
0x 60 05 00 01   IMM   R5,   0x1
0x 08 00 00 02   STORE R0,   R2
0x 20 00 00 03   ADD   R0,   R0, R3
0x 20 04 04 05   ADD   R4,   R4, R5
0x 11 20 04 01   BNE   0x20, R4, R1
```

Requirement of Instruction Set

- Instruction set, also called ISA (instruction set architecture) tells the CPU what needs to be done. It is directly related to the hardware structure and user needs.
- Key factors of a good instruction set:
  - Completeness

    Commonly used instructions are available, easy for programming
  - Effectiveness

    Require less resources, time, memory, higher execution speed

    - Easy operation

    Operating rules easy to learn and remember
  - Compatibility

    Program used in same series low-end model can run in hi-end model

CISC & RISC

- There are two major architectures:
  - CISC（complex instruction set computer）
    - One of the famous system is X86, developed by Intel
  - RISC（Reduced instruction set computer）
    - Arm is one of the major players of RISC, and the major market is embedded system

CISC	RISC
Hardware centric design Aim to do more by hardware in an efficient way	Software centric design Simpler hardware, less programming complexity and faster per instruction
Complex and variable length instructions	Simple, standardized instructions
Large number of instructions	Small number of fixed-length instructions
Require minimum amount of RAM / register	Require more RAM / register

Relationship between low level language and hardware

- Low level language – Machine language, Assembler
  - Machine originated, directly related to instruction set architecture
- High level language – more "user" friendly.

BASIS FOR COMPARISON	HIGH-LEVEL LANGUAGE	LOW-LEVEL LANGUAGE
Basic	Programmer amiable	Machine-friendly
Speed of execution	Fast	Slow
Translation	Requires compiler or an interpreter.	Assembler is required while machine language is directly executed.
Memory efficiency	Low	High
Comprehensibility	Understandable	Hard to understand
Portability and machine dependency	Portable and runnable in any platforms.	Non-portable and machine dependent.
Debugging and maintenance	Simple	Quite complex

Instruction set format

Opcode	Operand

- **Opcode** is the instruction tells the process what should be done
- **Operand** is the information that to be performed, can be data and/or memory location
- Both Opcode and Operand can be "length variate"
  - Machine dependence
  - Fixed length opcode can operate faster, good for small instruction set or for some frequent use command
  - Variable length opcode operation more complicate, but flexible and good for large system and better use hardware and software resources

Type of Instruction

- **Opcode**
  - Logical, Arithmetic
    - Add, subtract, And, Or…
  - Memory / register operation
    - Set / change specific function
  - Control transfer
    - Program branch, subroutine jump…
  - Complex
    - Some complex operation, hardware related microcode
- **Operands**
  - Address (register, RAM…)
  - Number (integer, floating point, binary, decimal… depend on system)
  - Character (coded character, e.g. ASCII)
  - Logical data (e.g. True, false)

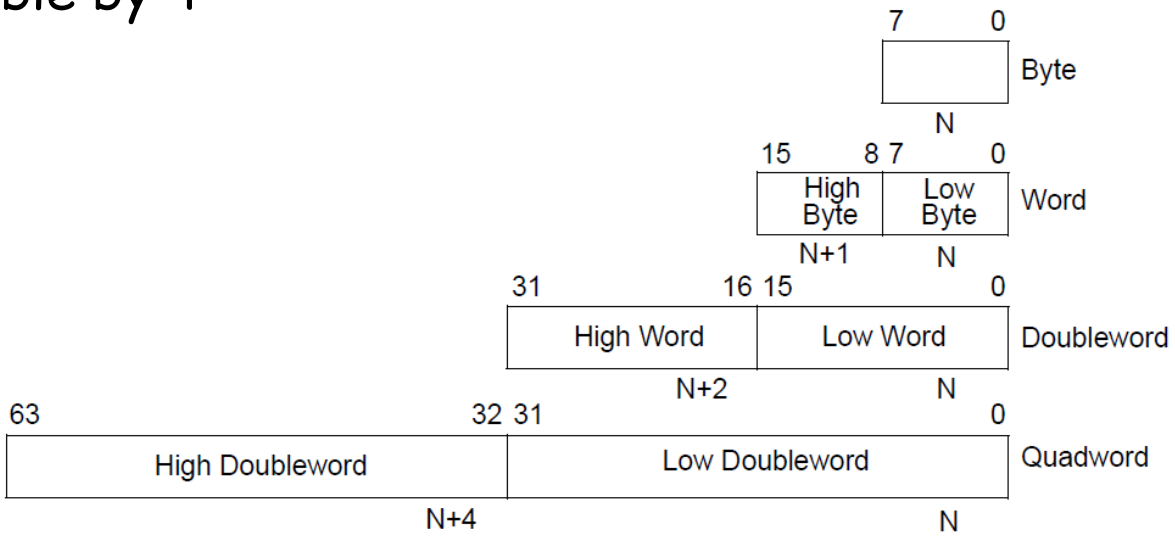
Variable length instruction Address

An instruction has 1 Opcode and may have several Operand

- No address (no Operand)
  - `CLC ;reset Carry flag to 0`
- 1 address instruction (direct address)
  - `MOV AL, 030H ;move 30H into AL register`
- 2 address instruction (Register indirect address)
  - `MOV AX, [BX] ;move the contents of memory location addressed by the register BX to the register AX`
- 3 address instruction (Another indirect address)
  - `MOV AX, [BX+SI+2] ;move the contents of memory location addressed by the content of register BX+ content of register SI +2, to the register AX`
- Multi-address instruction (4 or more address)

X86 (Intel) Data types

- 8 bit Byte
- 16 bit word
- 32 bit double word
- 64 bit quad word
- 128 bit double quadword
- Addressing is by 8 bit unit
- Words do not need to align at even-numbered address
- Data accessed across 32 bit bus in units of double word read at addresses divisible by 4
- Little endian

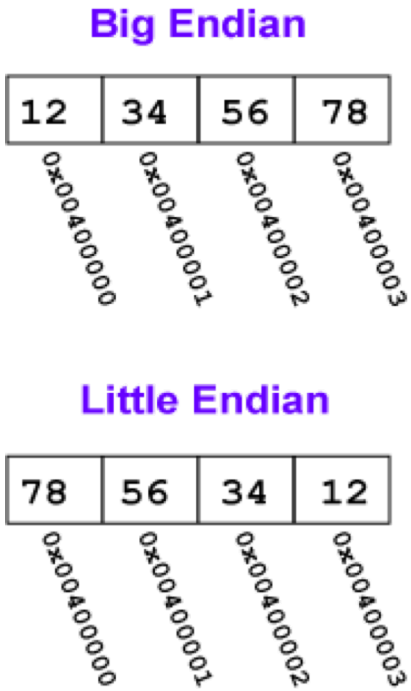


Byte Order Names

Endian - is the order or sequence of bytes of a word of digital data in computer memory.

- Stores the most significant byte of a word at the smallest memory address and the least significant byte at the largest. This is called big-endian
- Stores the least-significant byte at the smallest address.

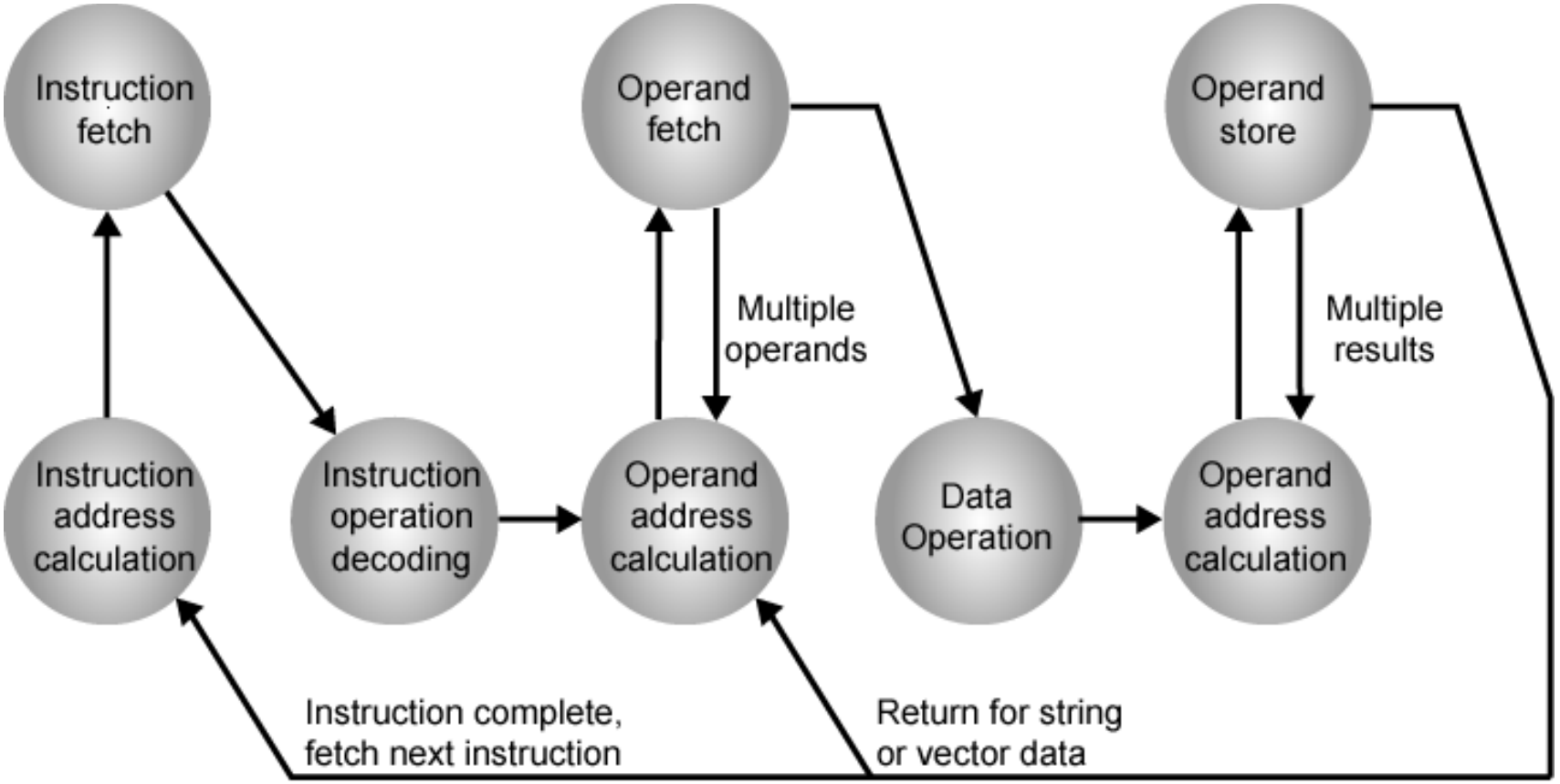
This is called little-endian



- Pentium (x86), VAX are little-endian
- IBM 370, Motorola 680x0 (Mac), and most RISC are big-endian
- Internet is big-endian
  - WinSock provides htoi and itoh (Host to Internet & Internet to Host) functions to convert

Danny Cohen introduced the terms *big-endian* and *little-endian* into computer science for data ordering in an Internet Experiment Note published in 1980

Instruction Cycle State Diagram



Exercise For Reader

- Find out about instruction set for Pentium and ARM
- Visit the following web sites

- https://developer.arm.com/documentation/100076/0100/a64-instruction-set-reference
- https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf