Introduction to Computer Architecture

Textbook Section 3.1-3.3

Computer Architecture

- Computer **organization** = hardware design:
  - the structure of a computer’s functional components
  - the functionality of a computer’s functional components
  - interconnection of a computer’s functional components
- Computer **architecture** = the set of resources provided by hardware for software
- This course is primarily about architecture & low-level programming
  - Some basic knowledge of the former is required.

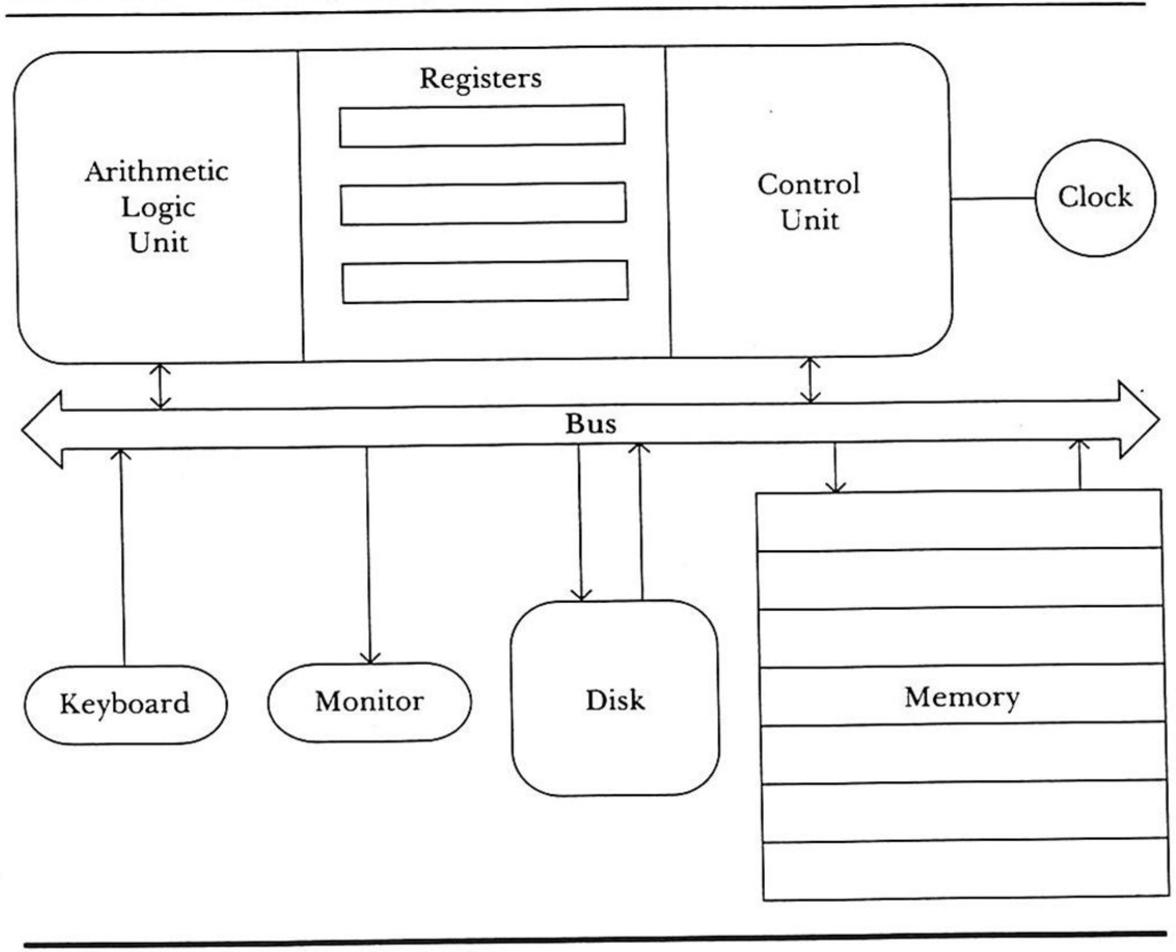
Von Neumann Architecture

- Memory is organized into **addressable** read/write storage locations.
- Memory contains **both** instructions and data
- By default, program execution proceeds **sequentially**, instruction by instruction
- A CPU **controls** this sequence and executes each instruction in turn
- The Harvard Architecture is an alternative to the Von Neumann Architecture
    - Code & data are stored in separate memories
    - Changing data cannot corrupt code **but**
    - Need twice as much memory/memory wastage

Computer Organization

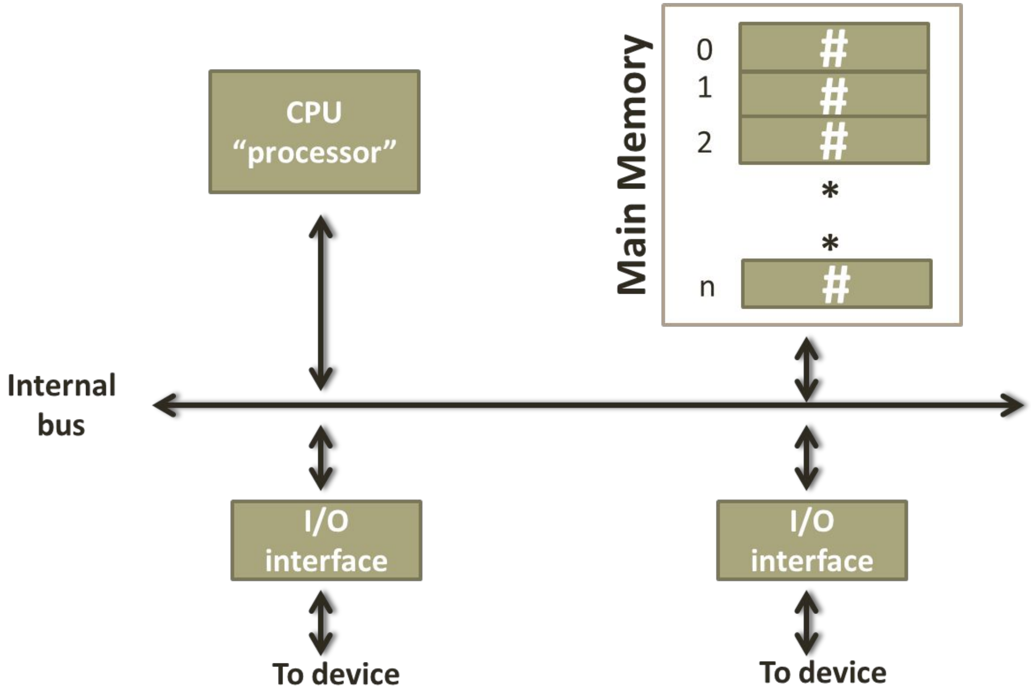
- I/O devices will be discussed in 2659
  - read section 3.2 anyway
- Other components will be discussed in detail

FIGURE 3.2 Basic Computer System



Computer Architecture

Hardware Component:	Architectural Resources Provided:
central processing unit(s) ("CPU" or "processor")	registers, instruction set, addressing modes
main memory	address space
I/O interfaces	(ignored for today)
bus, clock, etc.	communication



Address Spaces

Main memory

- On the motherboard, but **not** on the CPU
- Memory is organized into addressable, fixed-size storage locations for binary numbers

Address space

- Set of memory locations that can be read from or written to by the processor
- The processor must supply the address to access a memory location

Memory in the MC68000

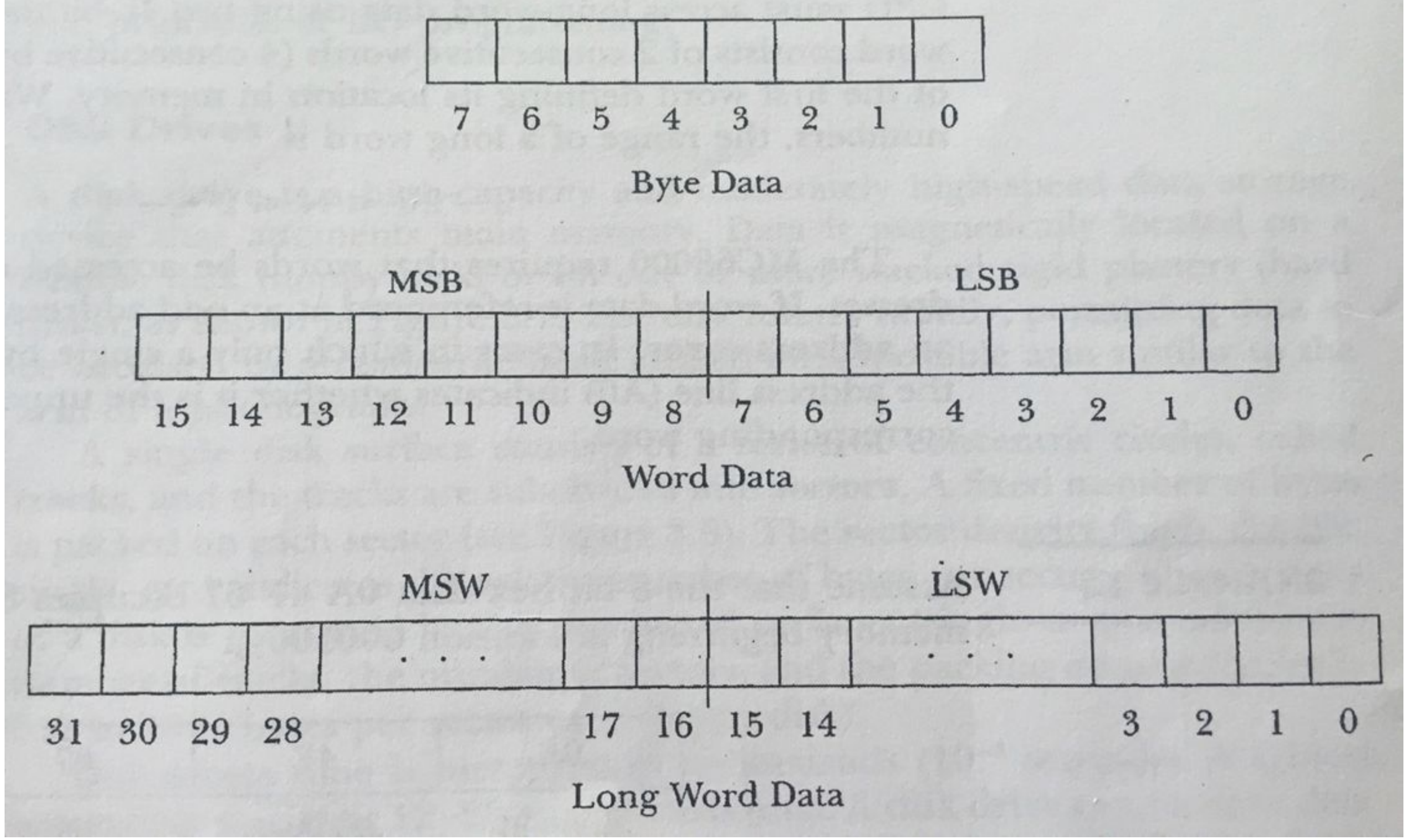
- If the smallest addressable unit of memory is **M**
    - The architecture is **M addressable**
- The 68000 architecture is **byte** addressable (i.e. M = byte)

```
    byte = 8-bits
    word = 16-bits
longword = 32-bits
```

- Caution: **word** is a relative term. For this course it will mean a 16-bit location

Byte, Word, Long Word

FIGURE 3.14 Byte, Word, and Long Word Data Organization



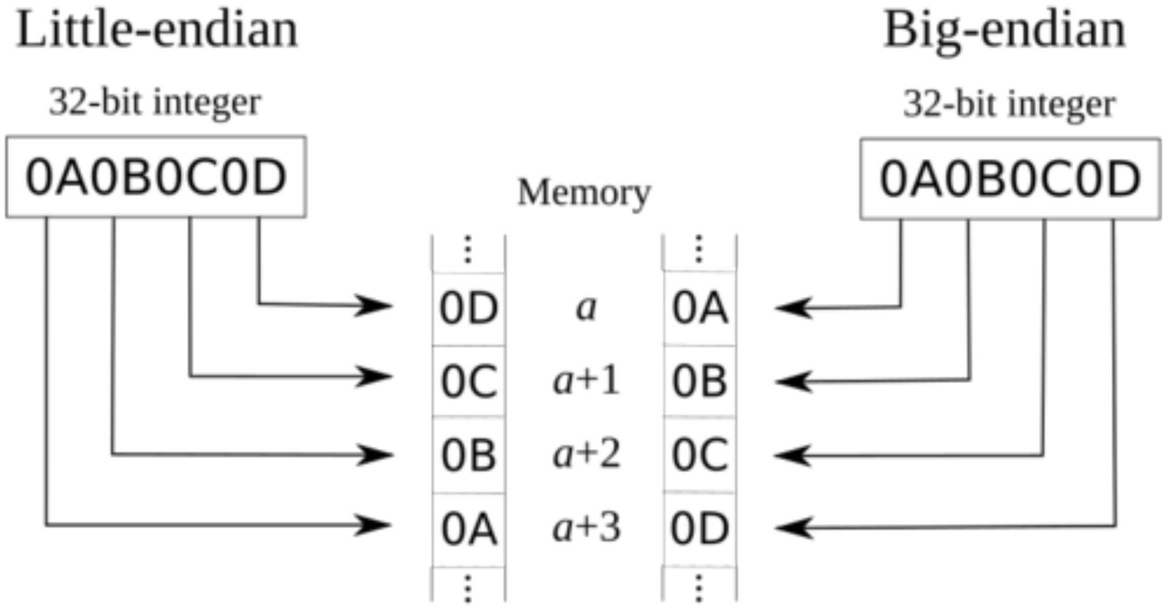
Endianness - Little Endian vs Big Endian

- Endian describes the order in which the byte of a multi-byte memory cell are stored
- The bits within a particular byte are always with the MSBit at the largest bit number
- Little Endian stores the Least Significant Byte (LSB) at the littlest address
    - Example: Intel Architecture
- Big Endian stores the LSB in the biggest address
    - Example: MC68000

Endianness

```
Little Endian (Intel)
address byte order
   MSB                        LSB
0  byte 3   byte 2   byte 1   byte 0
4  byte 7   byte 6   byte 5   byte 4

Big Endian (MC68000)
address byte order
   MSB                        LSB
0  byte 0   byte 1   byte 2   byte 3
4  byte 4   byte 5   byte 6   byte 7
```



Address Space

Address (In Base-16)	Data	
FFFFFF$_{16}$	10010011$_{2}$	“High Memory”
FFFFFE$_{16}$	11110000$_{2}$	
FFFFFD$_{16}$	01011010$_{2}$	
	. .	
4C60A0$_{16}$	7F$_{16}$	
	. .	
000001$_{16}$	2E$_{16}$	
000000$_{16}$	60$_{16}$	“Low Memory”

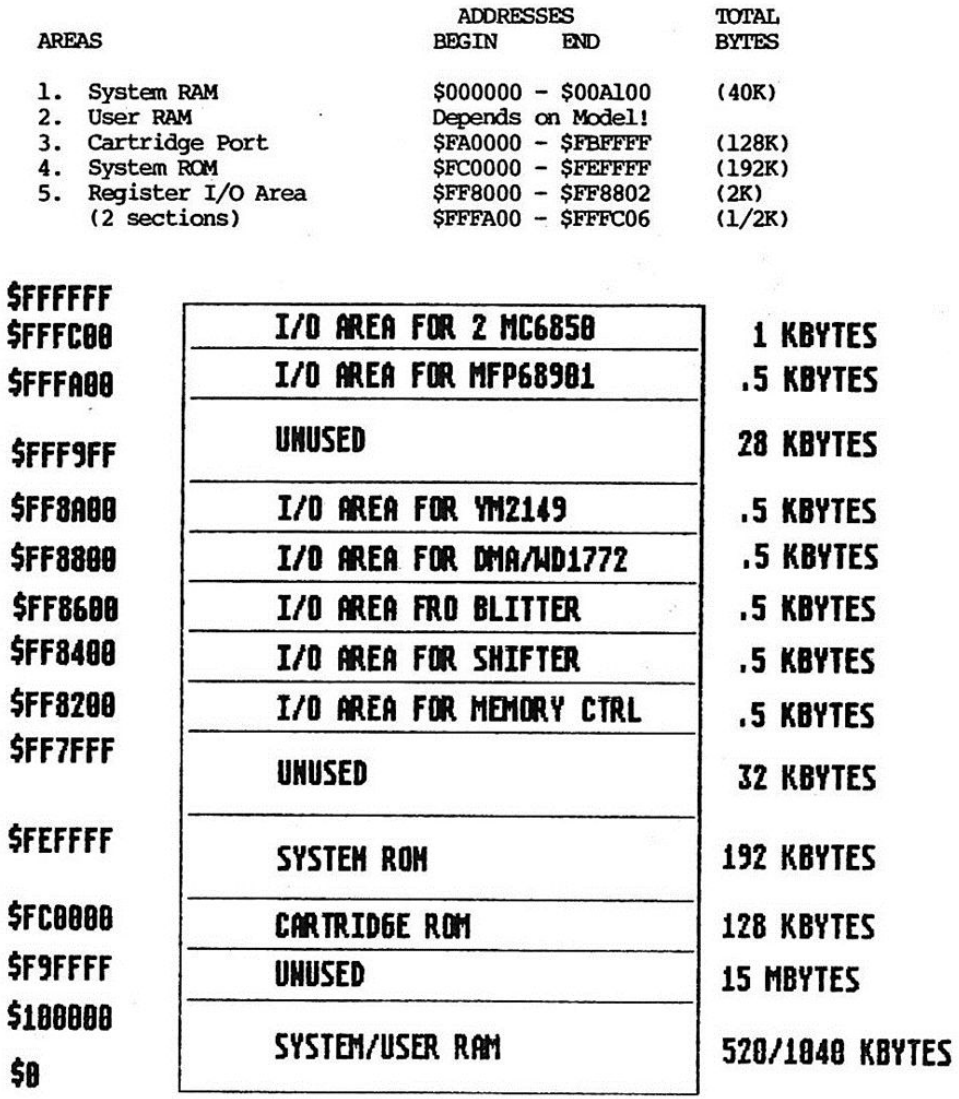
- The 68000 has an address space of 2^{24} bytes
  - 2^{24} = 16,777,216 bytes = 16M
- 32-bit architecture has address space of 2^{32} bytes
  - 2^{32} bytes = 4,294,967,296 = 4G
- 64-bit architecture has address space of 2^{64} bytes
  - 2^{64} bytes = WAY MORE!

MC68000 - Extra Details

- Bytes can be addressed at any address
  - Example: Can address a byte at address 255_{10}
- Words and longwords can **ONLY** be addressed at even addresses
  - Example: Cannot address either word or longword at address 255_{10}
  - Example: Can address the byte, word and longword at address 256_{10}

Atari Memory Map

While the 68000 has an address space of 16M, the Atari only allows 4M of physical memory

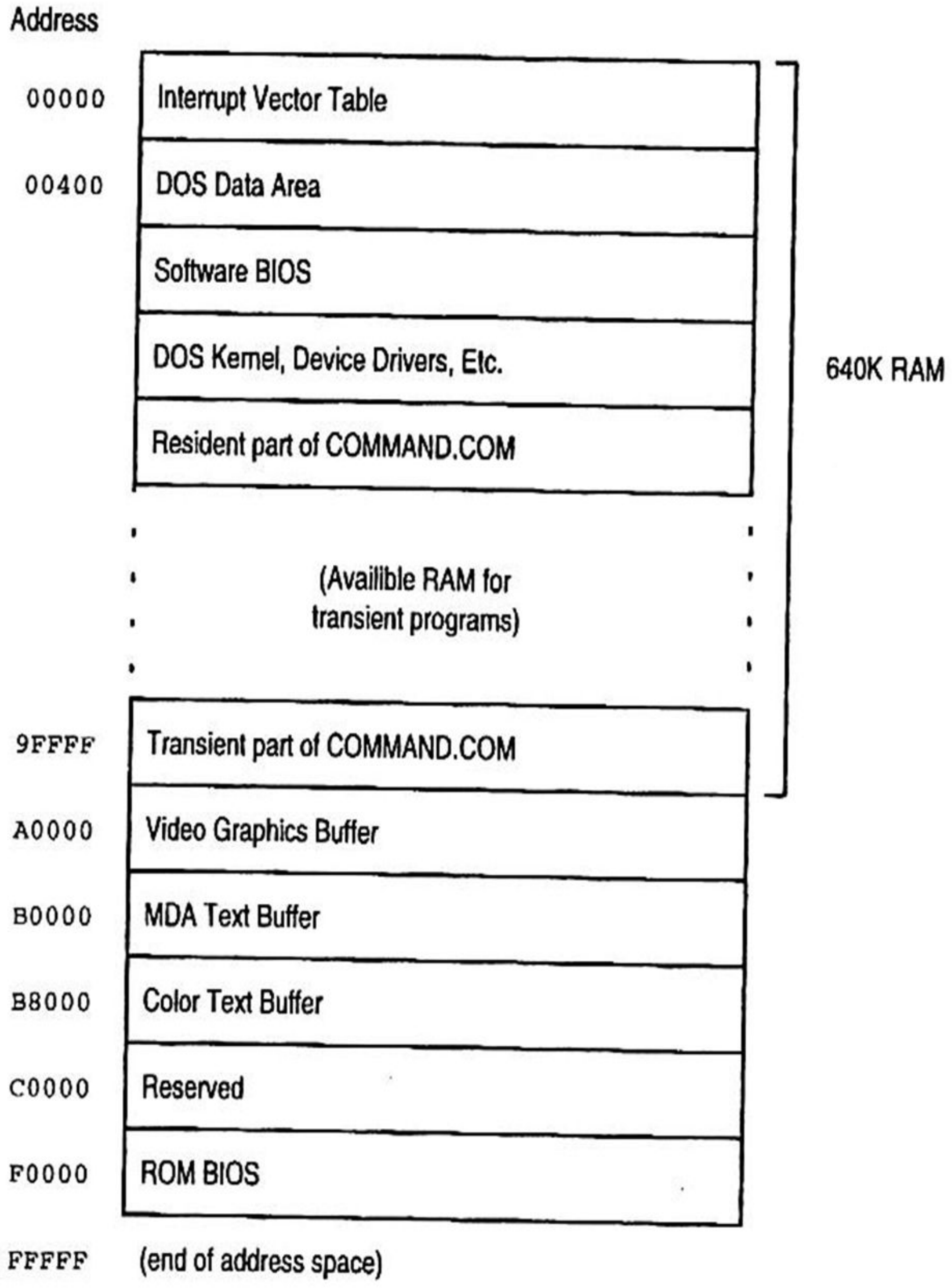


AREAS	ADDRESSES BEGIN	 END	TOTAL BYTES
1. System RAM	$000000 -	$00A100	(40K)
2. User RAM	Depends on	Model!	
3. Cartridge Port	$FA0000 -	$FBFFFF	(128K)
4. System ROM	$FC0000 -	$FEFFFF	(192K)
5. Register I/O Area	$FF8000 -	$FF8802	(2K)
(2 sections)	$FFFA00 -	$FFFC06	(1/2K)

$FFFFFF $FFFC00	I/O AREA FOR 2 MC6850	1 KBYTES
$FFFA00	I/O AREA FOR MFP68901	.5 KBYTES
$FFF9FF	UNUSED	28 KBYTES
$FF8A00	I/O AREA FOR YM2149	.5 KBYTES
$FF8800	I/O AREA FOR DMA/WD1772	.5 KBYTES
$FF8600	I/O AREA FRO BLITTER	.5 KBYTES
$FF8400	I/O AREA FOR SHIFTER	.5 KBYTES
$FF8200	I/O AREA FOR MEMORY CTRL	.5 KBYTES
$FF7FFF	UNUSED	32 KBYTES
$FEFFFF	SYSTEM ROM	192 KBYTES
$FC0000	CARTRIDGE ROM	128 KBYTES
$F9FFFF	UNUSED	15 MBYTES
$100000 $0	SYSTEM/USER RAM	520/1040 KBYTES

Details of First MB of PC Memory

Memory is partitioned into special-purpose regions or **segments**



Address	
00000	Interrupt Vector Table
00400	DOS Data Area
	Software BIOS
	DOS Kernel, Device Drivers, Etc.
	Resident part of COMMAND.COM
	(Availible RAM for transient programs)
9FFFF	Transient part of COMMAND.COM
A0000	Video Graphics Buffer
B0000	MDA Text Buffer
B8000	Color Text Buffer
C0000	Reserved
F0000	ROM BIOS
FFFFF	(end of address space)

640K RAM

Recap

- RAM
  - Random Access Memory (read/write, volatile)
- ROM
  - Read-Only Memory (read, non-volatile, contains BIOS etc.)
- Binary Powers
  - 2^{10} = 1,024 = 1K
  - 2^{20} = 1,048,576 = 1M = 1,024K
  - 2^{30} = 1,073,741,824 = 1G= 1,024M

Recall

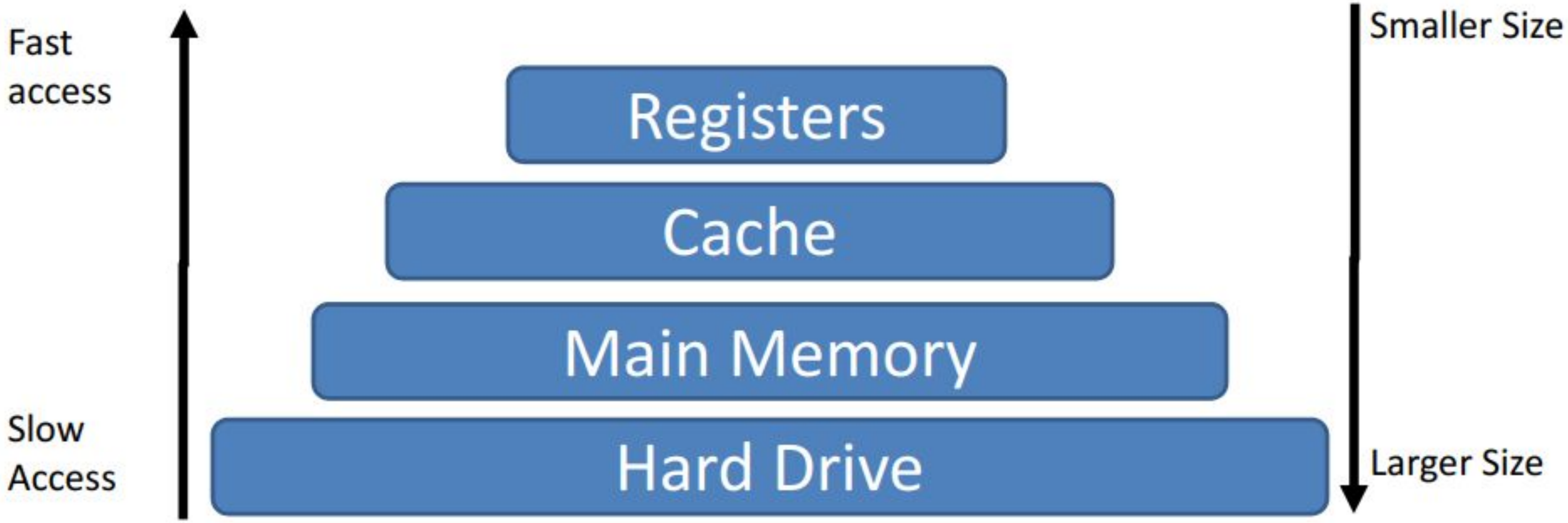
What is the

- least significant bit?
- the sign bit?
- least significant byte?
- most significant byte?
- least significant word?
- most significant word?
- low order bits/byte/word?
- high order bits/byte/word?

Additional Details

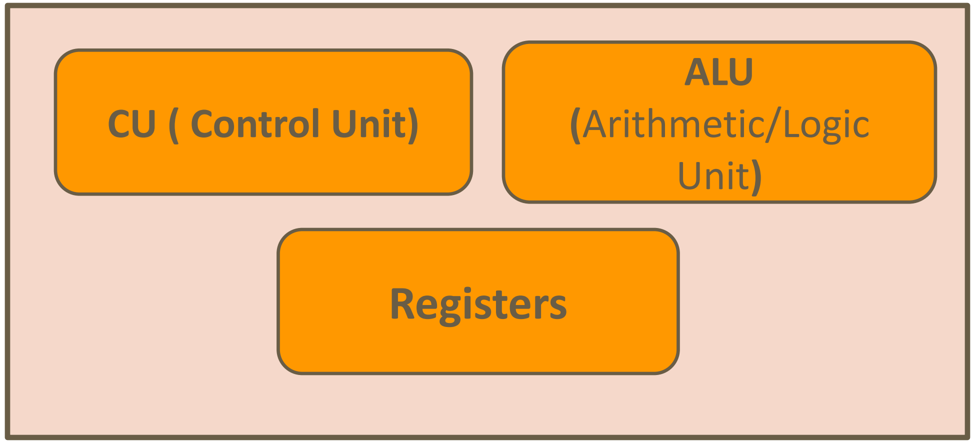
- An **executable program** is a sequence of machine language instructions (+ data).
- Modern computers are stored program computers:
    - Memory **contains data**!
    - Memory also **contains code**!
    - To run a program, the O/S must:
        - **Load the program** from disk into memory (or some portion of it)
        - **Jump** to the first instruction in sequence
    - CPU **can not** distinguish between data and code!

Memory


Fast access
Slow Access
Registers
Cache
Main Memory
Hard Drive
Smaller Size
Larger Size

CPU - Simplified View

CU - The CU is the circuit that directs all other CPU activity, like a conductor.



ALU - The ALU is a “library” of arithmetic & logic circuits.
E.g.

- adder circuit, etc.
- bitwise AND, OR, XOR, etc. circuits
- etc.

The CPU delegates various arithmetic/logic operations to the ALU.

Register - a fast memory location on-board the CPU.

A register has a name, but not an “address” in the same sense a main memory location does.

CU - Control Unit

- The CU is the **conductor** of the CPU
  - it coordinates all CPU activity
- Specifically, the CU directs the instruction execution cycle. For each instruction in sequence:
  - Fetch phase - organizes fetch of instruction from memory
  - Decode phase - based on the **binary** content of memory determines the operation
  - Execute - fetches all operands and if ALU is involved delegates to it, handles storing result

ALU - Arithmetic Logic Unit

- The ALU is a **library** of arithmetic and logic circuits.
- E.g.
  - Adder (may be multiplier)
  - Bitwise AND, OR, XOR, etc. circuits
  - One’s complement negation circuit,
  - Etc.
- Each of these actions will have a corresponding machine (assembly) language instruction
- Additional instructions not using the ALU will also exist.

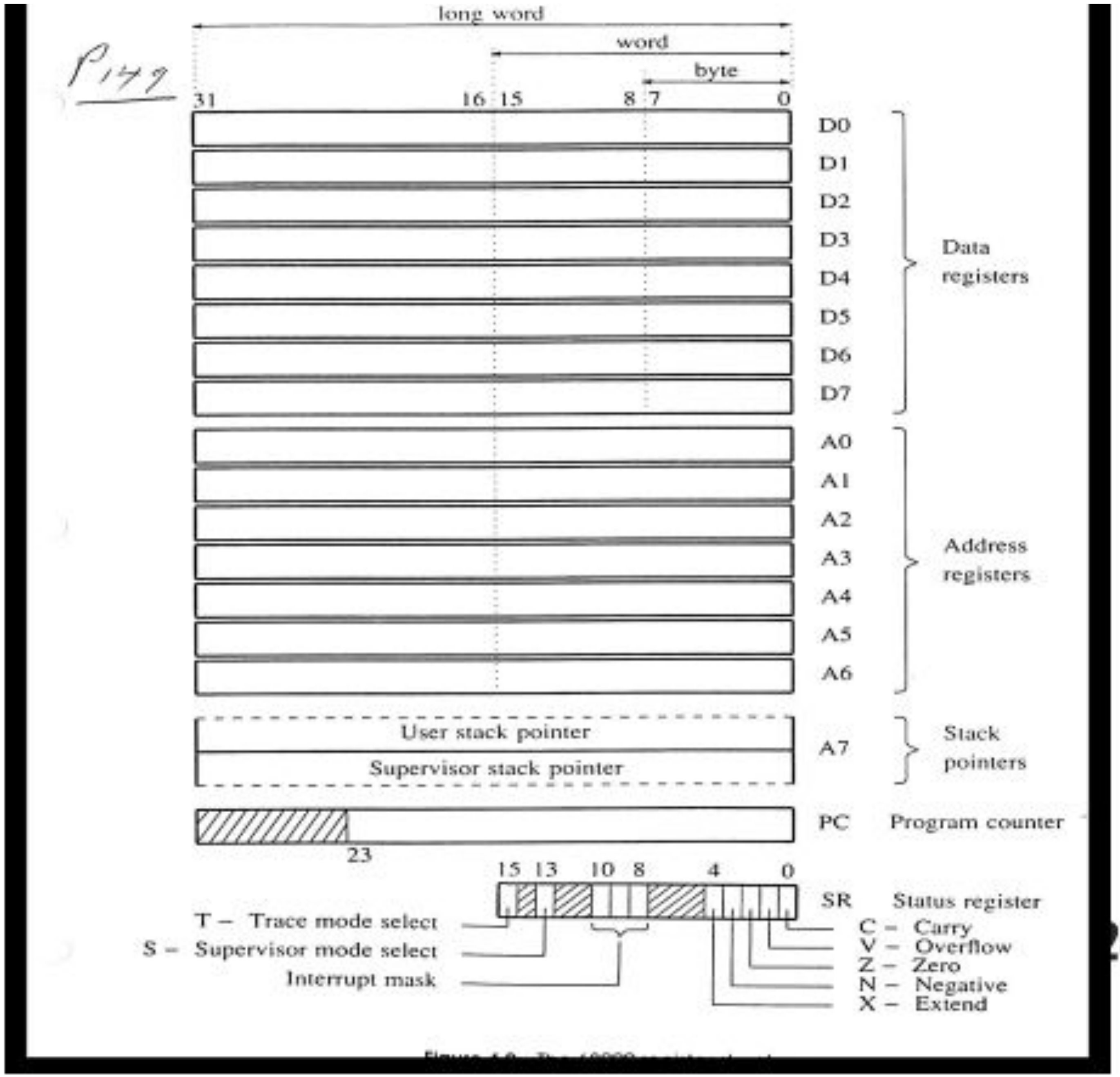
ALU - Arithmetic Logic Unit

- The ALU arithmetic/logic operations can be viewed as circuits, such as an adder
- These operations will have two results:
    - A direct result (e.g. the sum)
    - An indirect result (e.g. NZVC – status bits)
- When an ALU circuit is invoked, the status bits are often saved. Why?

Registers

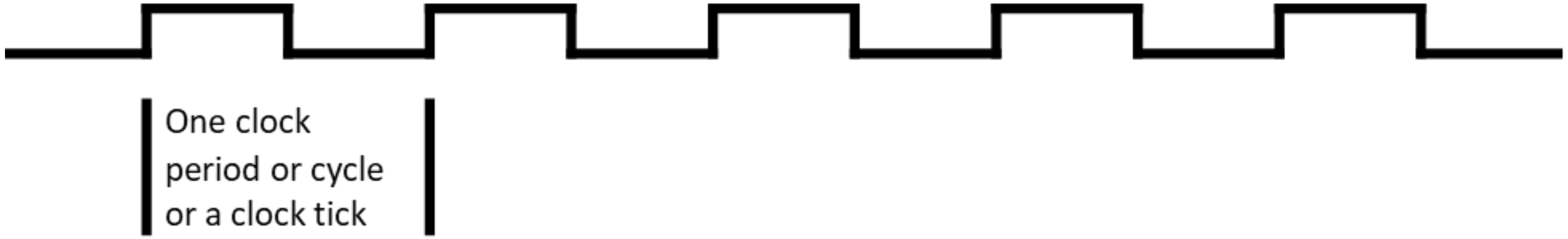
- A fast memory location on-board the CPU.
- A register has a name, but not an “address”.
- Two kinds of registers:
  - **General purpose** – programmer has complete access
  - **Special purpose** – programmer has limited or no access

Register Layout



The Clock

- The clock is the **pacemaker** or **heart** for the CPU and ultimately the entire computer
- Everything in a computer is done based on a clock signal
- The clock is a crystal which generates a regular pulse for timing system activity
- Frequency is how clocks are speeds referenced and they are measured in terms of Hertz – cycles per second
- A typical clock signal looks like this:



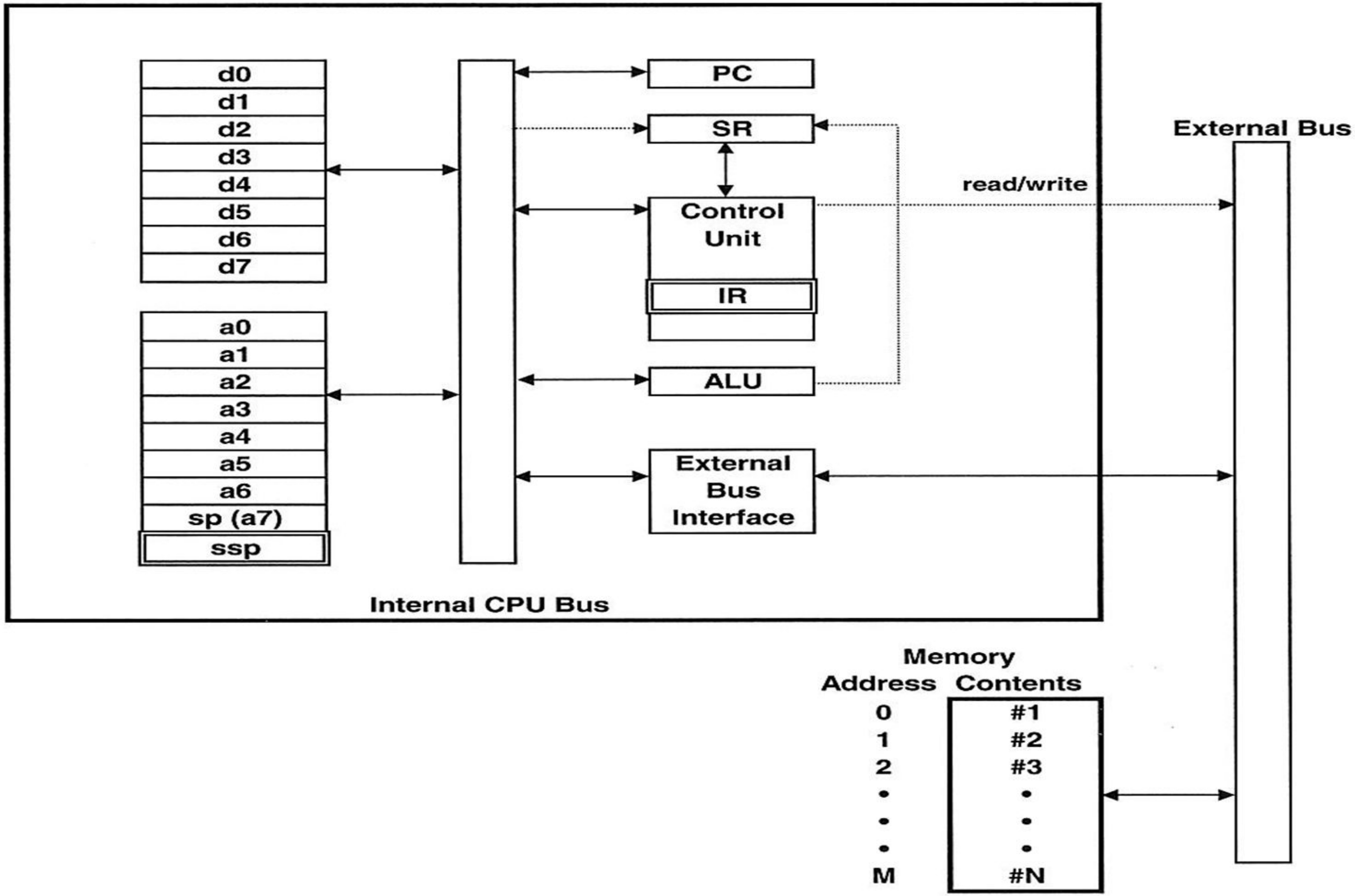
One clock period or cycle or a clock tick

- Period is the time for one clock tick, which is 1 / frequency
- Most instructions take more than one clock cycle to complete!
- Still this means computers work extremely fast!

A Bus

- Is a communication system that transfers data between components inside a computer.
- It consists of:
    - Address bus
    - Data bus
    - Control bus
- A computer will contain more than one bus
    - Required to communicate between different components
    - Not all buses will require information from all three bus components
        - Example: Read-only devices, Network devices

Idealized 68000 CPU



Address	Contents
0	#1
1	#2
2	#3
•	•
•	•
•	•
M	#N

Summary

- Computer Architecture
    - Von Neumann Architecture
    - Main Memory
    - CPU
        - Control Unit (CU)
        - Arithmetic Logic Unit (ALU)
        - Registers
    - Bus
    - Clock