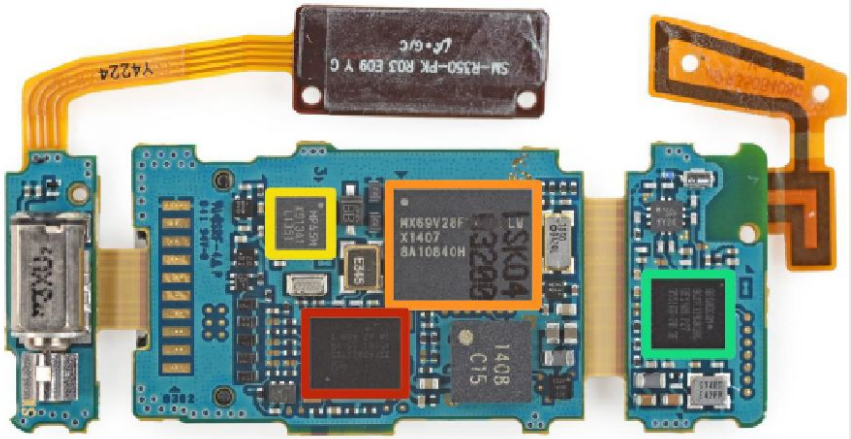
# **CSE331:** MICROPROCESSOR INTERFACING & EMBEDDED SYSTEMS

Lecture 2: Introduction to microprocessor #2
ARM processor

Mohammad A Qayum, PhD

# Samsung gear fitness tracker

- STMicroelectronics **STM32F4**39ZI 180 MHz, 32 bit **ARM Cortex-M4** CPU

# Iphone 4 breakdown

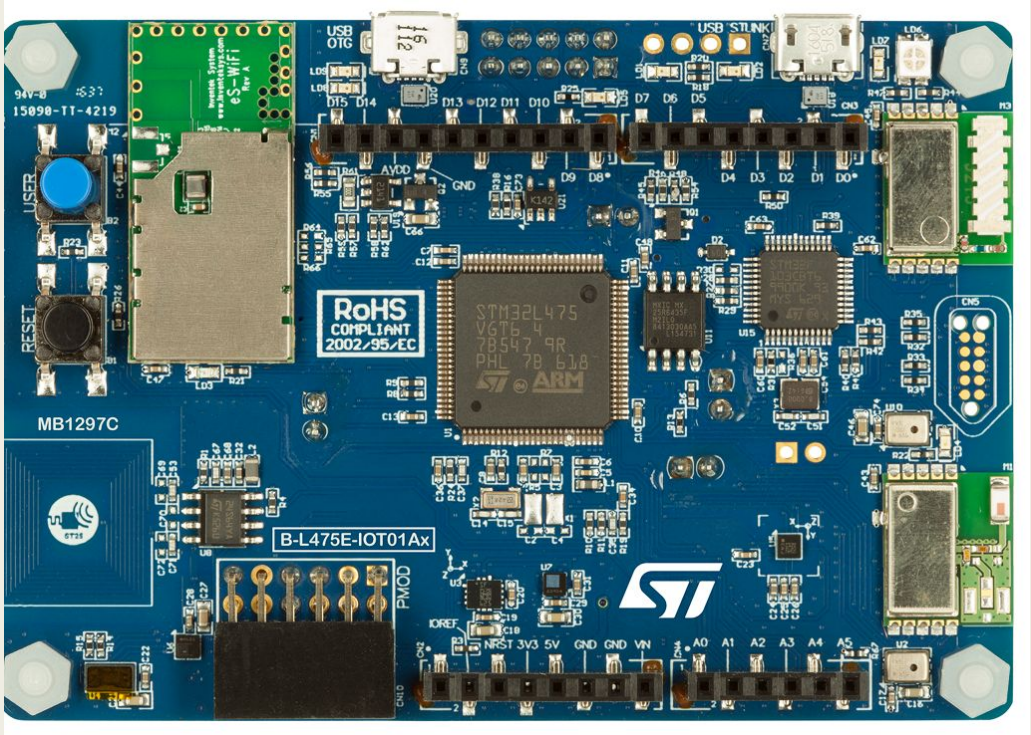

The A6 processor is the first Apple System-on-Chip (SoC) to use a custom design, based off the **ARMv7** instruction set. It is a predecessor to the **ARM Cortex-M4** we'll be using.

This is an iPhone 4. **What processor does the iPhone 8 use? Iphone X?**

# Features of ARM Cortex-M4 Microprocessor (STM32L475)

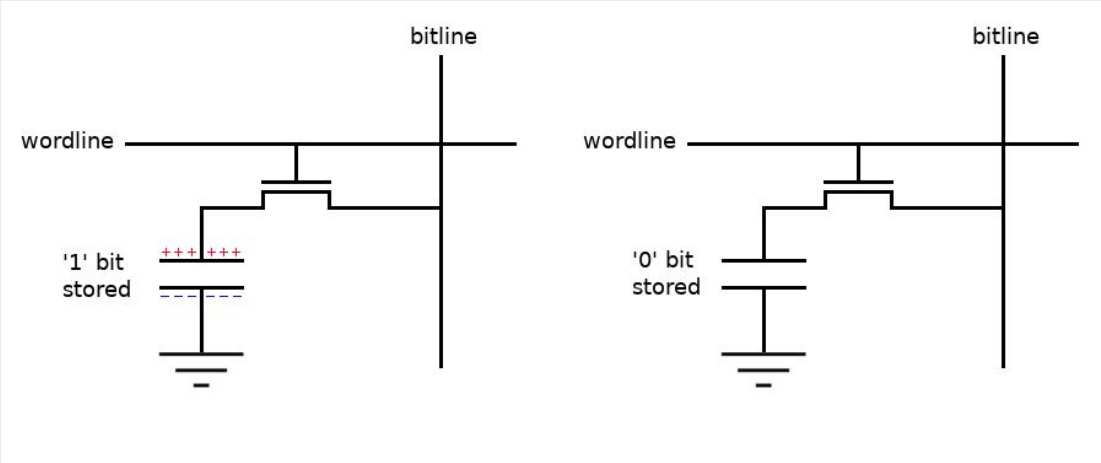
- 32-bit CPU (32 bit wide registers and address path)
- 856-915 MHz max internal clock
- 128 kBytes of on-chip RAM
- 1 MBytes of Flash memory
- Sophisticated timer functions that include: input capture, pulse output, …
- Communication interfaces: USART, SPI, I2C, CAN, …
- A/D and D/A converters

# STM32L4 in a development board

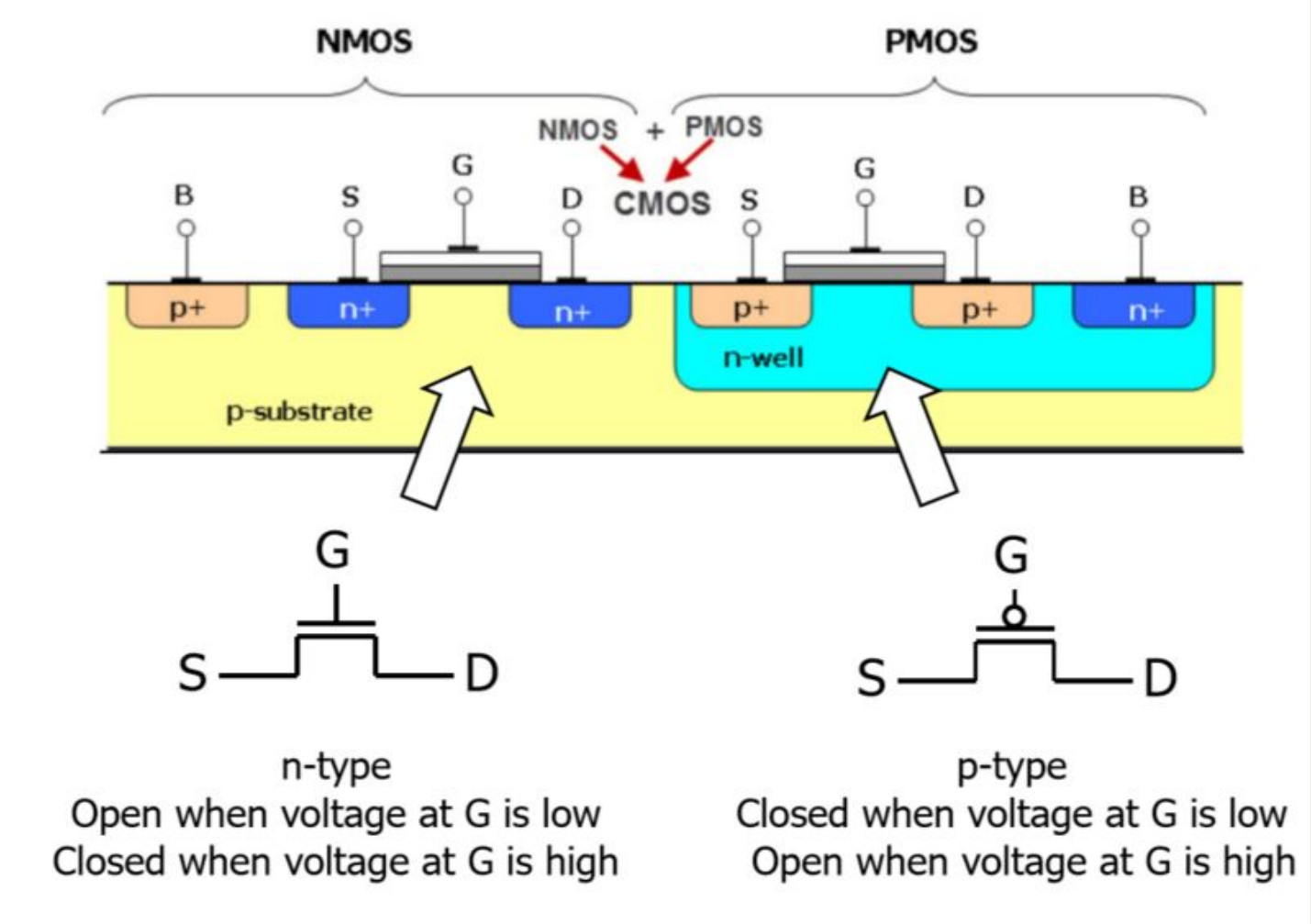


# Let us recap some basics- Electronics

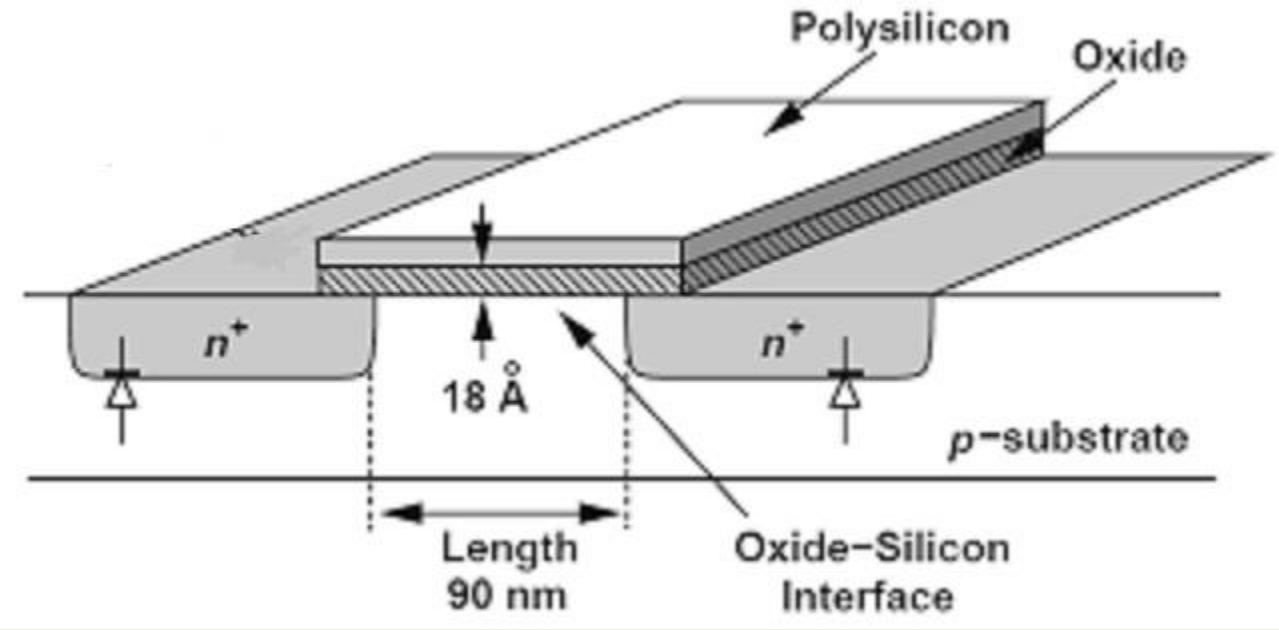
- Analog circuits (R. L, I) and Digital Circuits (Transistor)
- Flow of electrons is Current
- HI or “1” logic is 5 volt that means large amount of electron will flow from measuring point to common point
- LOW or “0” logic is 0 volt that means no amount of electron will flow from measuring point to common point
- “1” logic stores enough electrons through capacitance effect using semiconducting technology that will create 5 volt potential



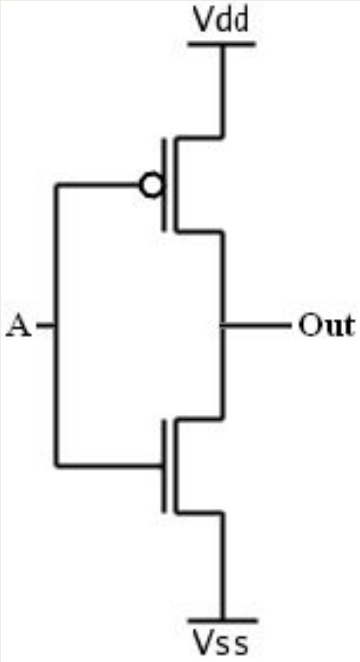
# MOSFET: nMOS & pMOS



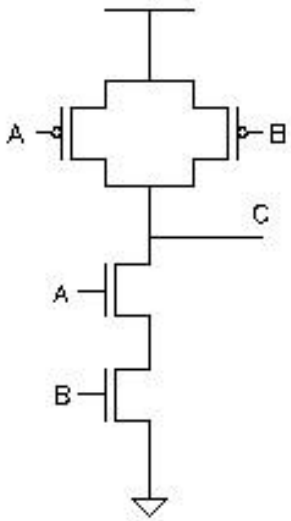
# MOSFET: Channel Length (process technology node- 90nm)



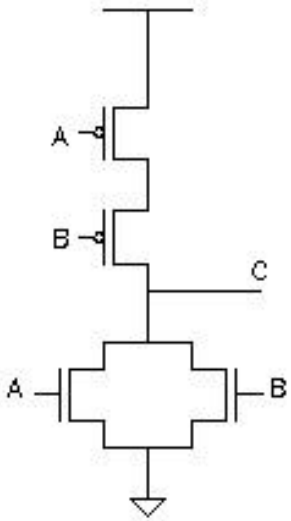
# CMOS: not, nand, nor




NOT




NAND




NOR

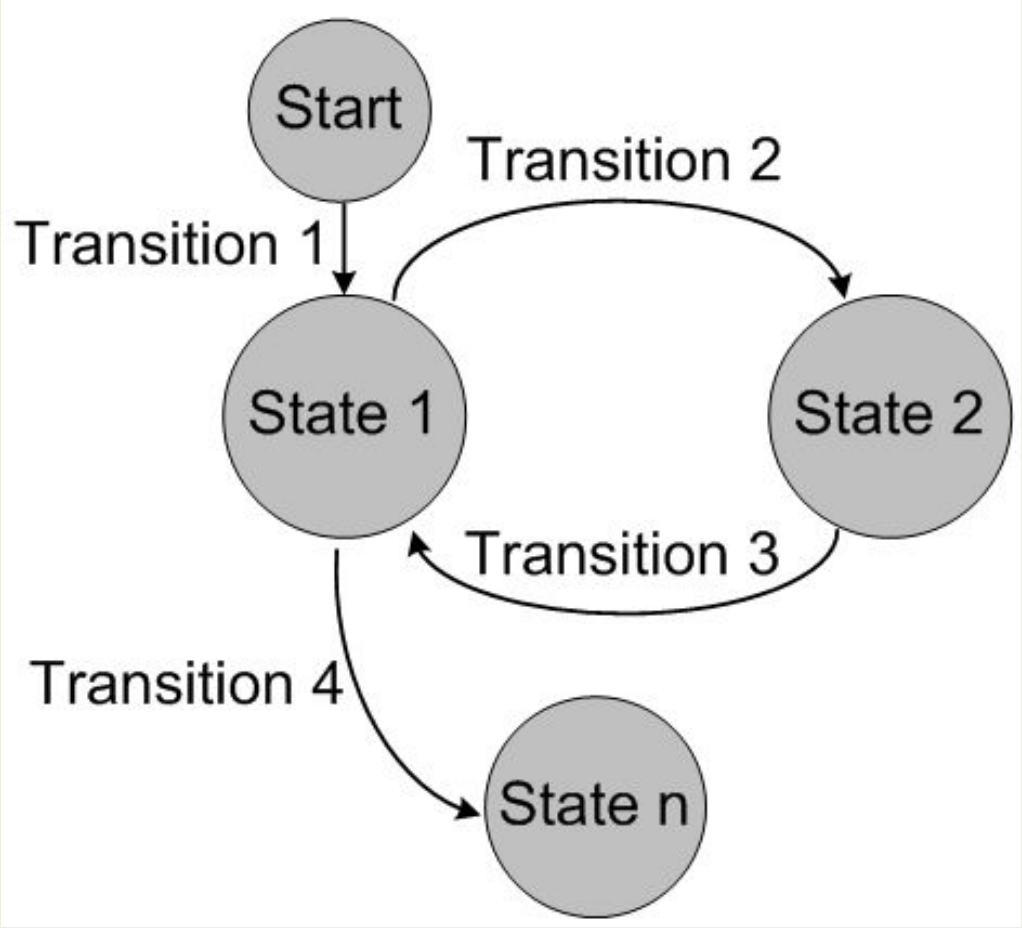
# Digital Circuit Classification: Combinational Circuits

- Output depends only solely on the current combination of circuit inputs
- Same set of input will always produce the same outputs
- Consists of basic gates such as AND, OR, NOR, NAND, and NOT gates

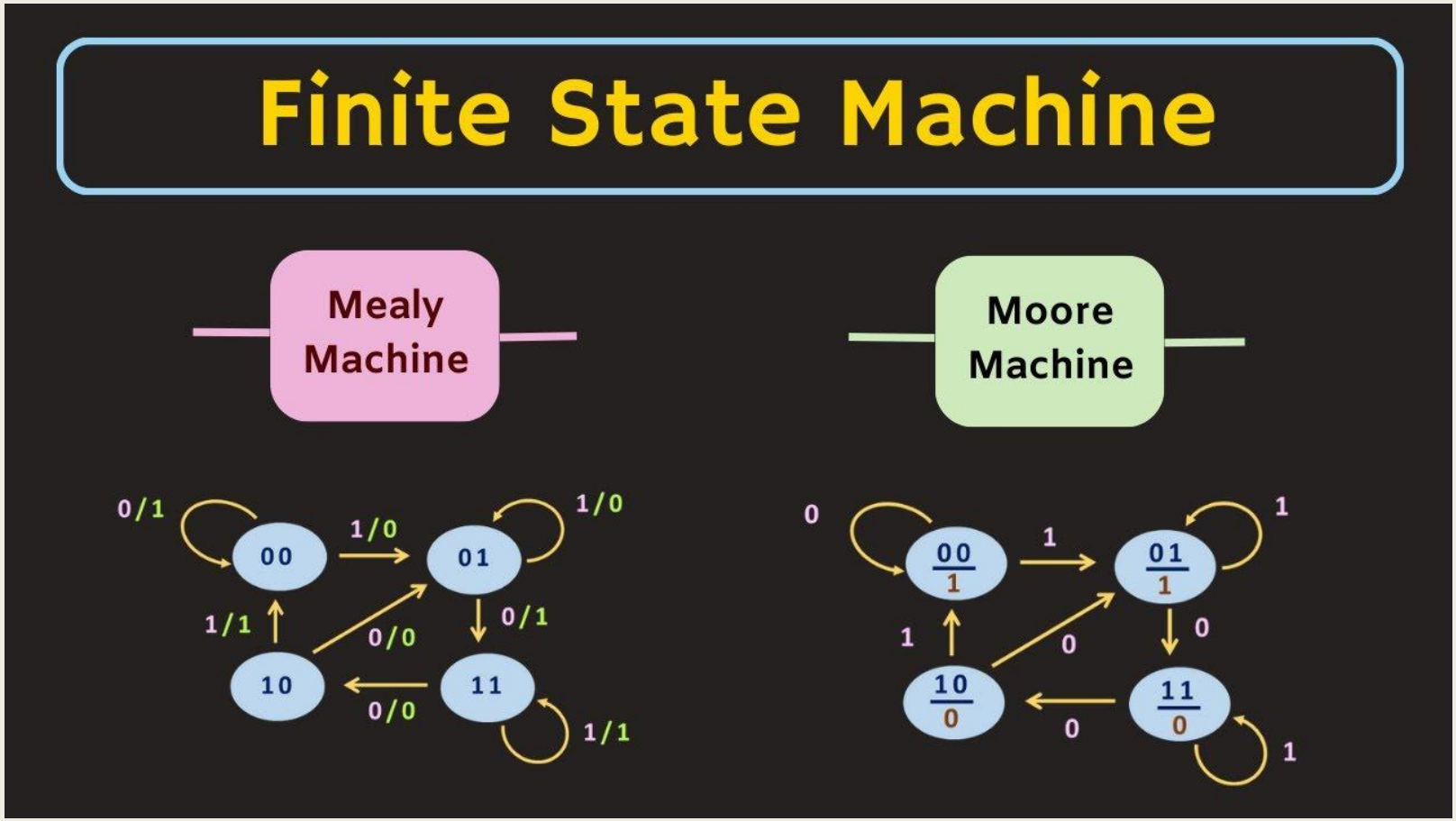
# Sequential Circuits

- *Output depends on the current inputs and state of the circuit (or past sequence of inputs)*
- *Memory elements such as flip-flops and*
- *registers are required to store the "state"*
- *Same set of input can produce completely different outputs*

# Sequential Circuit: State Machines

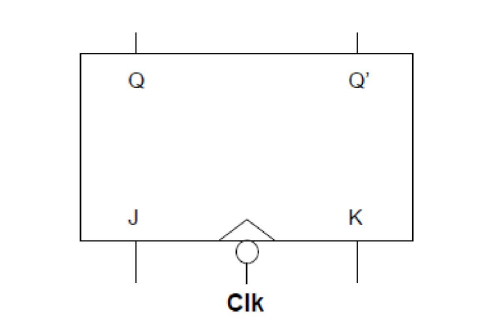


# Mealy and Moore Machine



# Flipflops- unit of memory

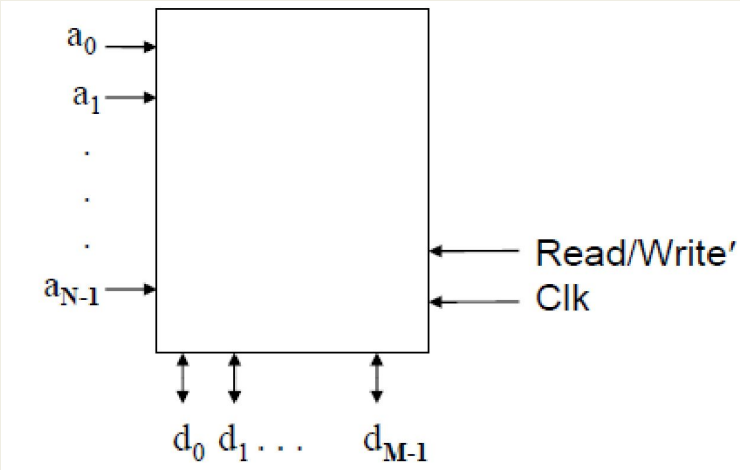
- A key memory device in microprocessors is the flip-flop.
- Remember: Flip-flops are clocked. Latches are not.
- Computers use clocked circuits => flip-flops.
- For example, a J-K flip-flop is shown as:




- In microprocessors, flip-flops are typically grouped together with common control signals into registers.

# Memory

- Another common subsystem in microprocessors is MEMORY, with multiple address lines and multiple data lines:

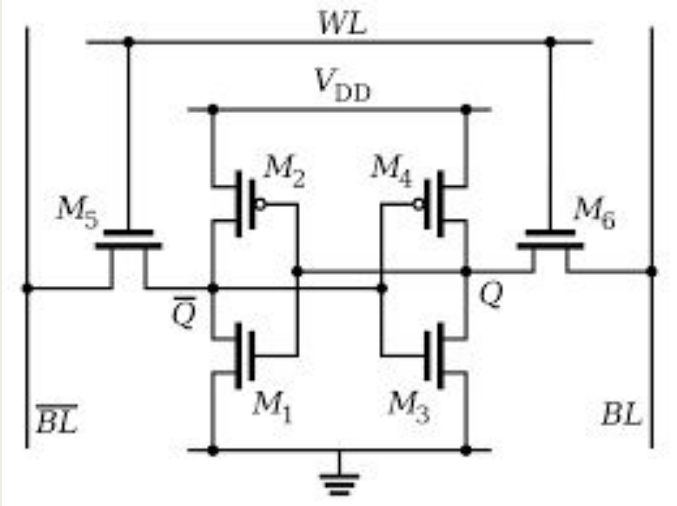



- The above has N address lines and M data lines. The capacity of this memory system is ???

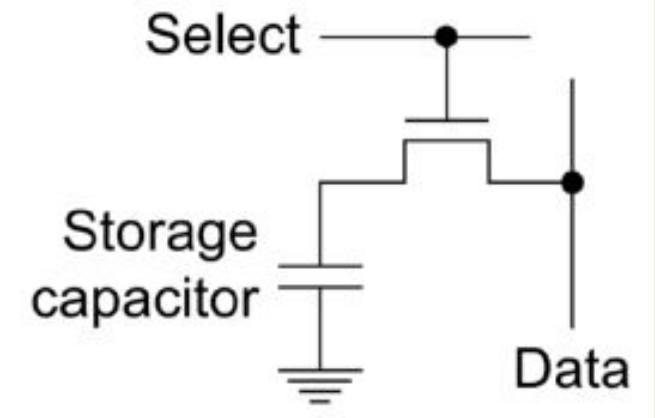
# Memory Types: ROM and RAM

- **Random Access Memory**
  - ***Dynamic Random Access Memory (DRAM):*** *periodic refresh is required to maintain the contents of a DRAM chip*
  - ***Static Random Access Memory (SRAM):*** *no periodic refresh is required. Always more predictable and usually faster than DRAM.*
- **Read-Only Memory**
  - ***Mask-programmed read-only memory (MROM):*** *programmed when being manufactured*
  - ***Programmable read-only memory (PROM):*** *the memory chip can be programmed by the end user*

# Memory: SRAM and DRAM




SRAM single cell




DRAM single cell

# EPROM, EEPROM, FLASH

- **Erasable Programmable ROM (EPROM)**
  1. Electrically programmable many times
  2. Erased by ultraviolet light (through a window)
  3. Erasable in bulk (whole chip in one erasure operation)
- **Electrically Erasable Programmable ROM (EEPROM)**
  1. Electrically programmable many times
  2. Electrically erasable many times
  3. Can be erased one location, one row, or whole chip in one operation
- **Flash Memory**
  1. Electrically programmable many times
  2. Electrically erasable many times
  3. Can only be erased in bulk

# Memory Addressing

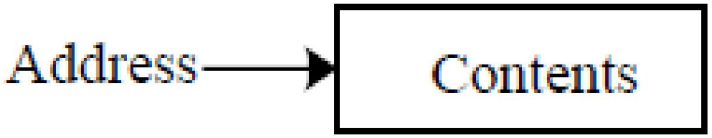
- **Memory Addressing**

  **-** Memory consists of a sequence of directly addressable locations.

  - A location is referred to as an information unit.

  - A memory location can be used to store data and instructions.

  - A memory location has two components: an address and its contents

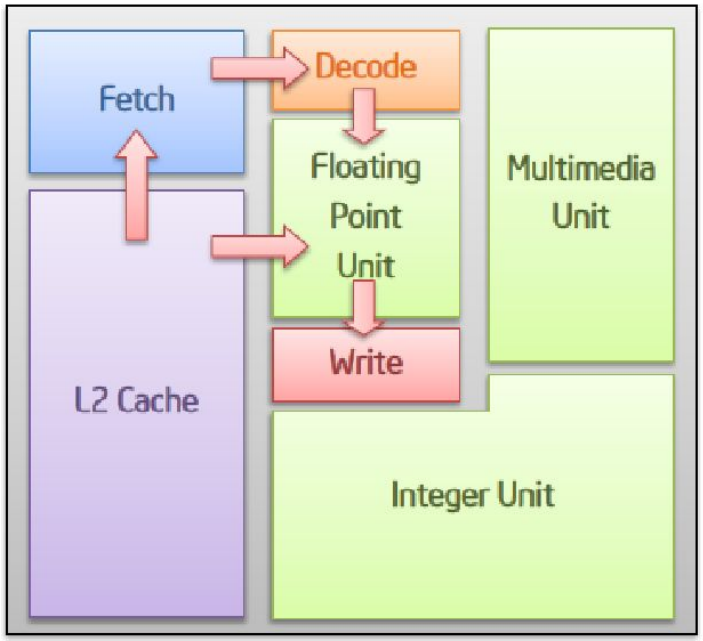



The components of a memory location

# Architecture Basics

- **Computer Programs -** Computer programs (programs) are known as software - A program is a sequence of machine instructions
- **Machine Instruction -** A sequence of binary digits that can be executed by the processor - Hard to understand, program, and debug for human being - E.g. in the ARM Cortex M4,
  - *instruction* ***0010*** *0010 0110* ***0100***
  - *How many bytes long is this instruction? Instruction is also 0x2264.*
- **Assembly Language -** Defined by machine instructions - An assembly instruction is a mnemonic representation of a machine instruction,
  - *ADD R2 R1 R0 represents 0001100 010 001 000 = 0x1888 (hex format)*

# CPU stages




Fetch Unit gets
the next instruction from the cache.

Decode Unit determines
type of instruction.

Instruction and data sent to
Execution Unit.

Write Unit stores result.

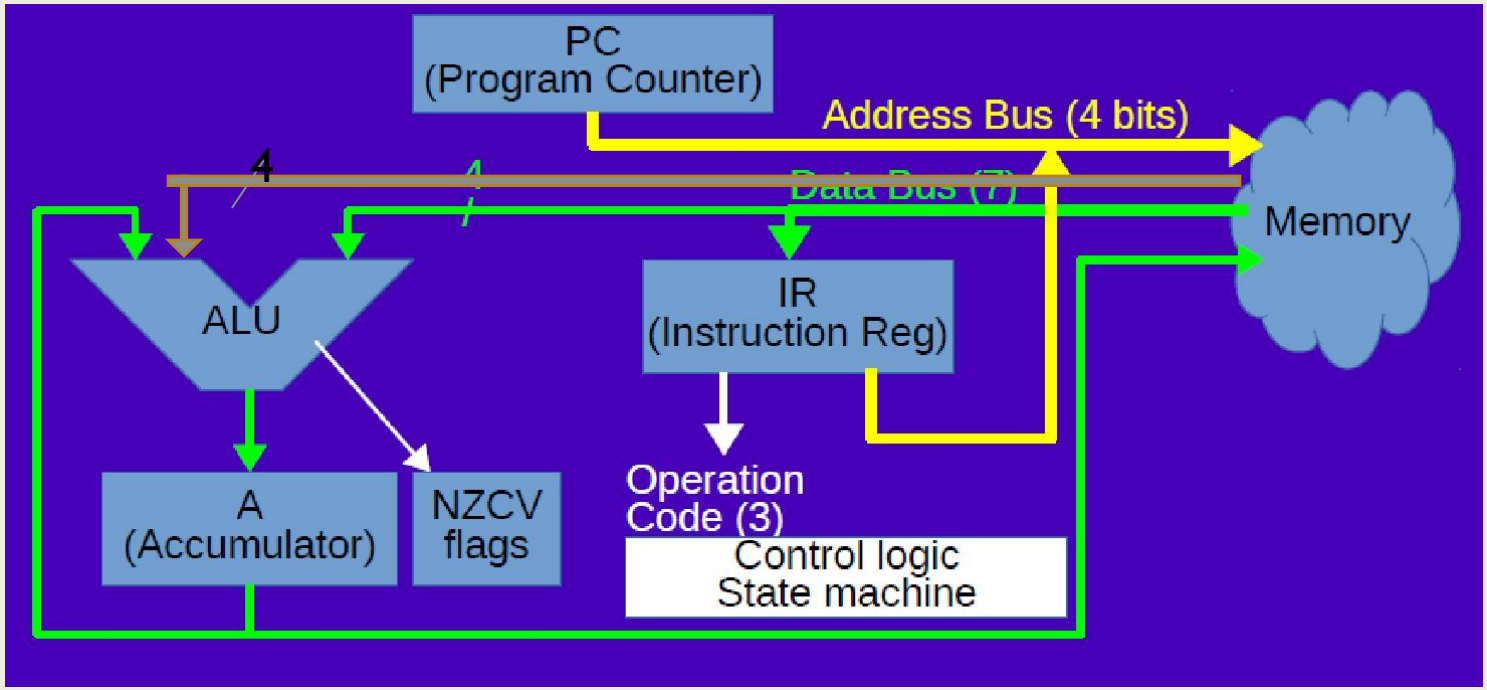
# Architecture and Operations

- **Program counter:** register PC “points to” (i.e. contains the address of) the memory location containing the next instruction to be executed.
- **Fetch:** Using the current value of PC, read the next instruction to execute from memory. Store this data in the instruction register, IR.
- Note that IR is an internal register, not accessible with computer instructions.

# CPU stages (Pipelines)

- **Decode :** Determine which instruction is to be executed using the contents of the IR.
- **Execute :** Activate appropriate control lines in processor sequentially to perform the action specified by the Instruction at particular clock ticks.
- **Write:** Write the output to the Data Memory

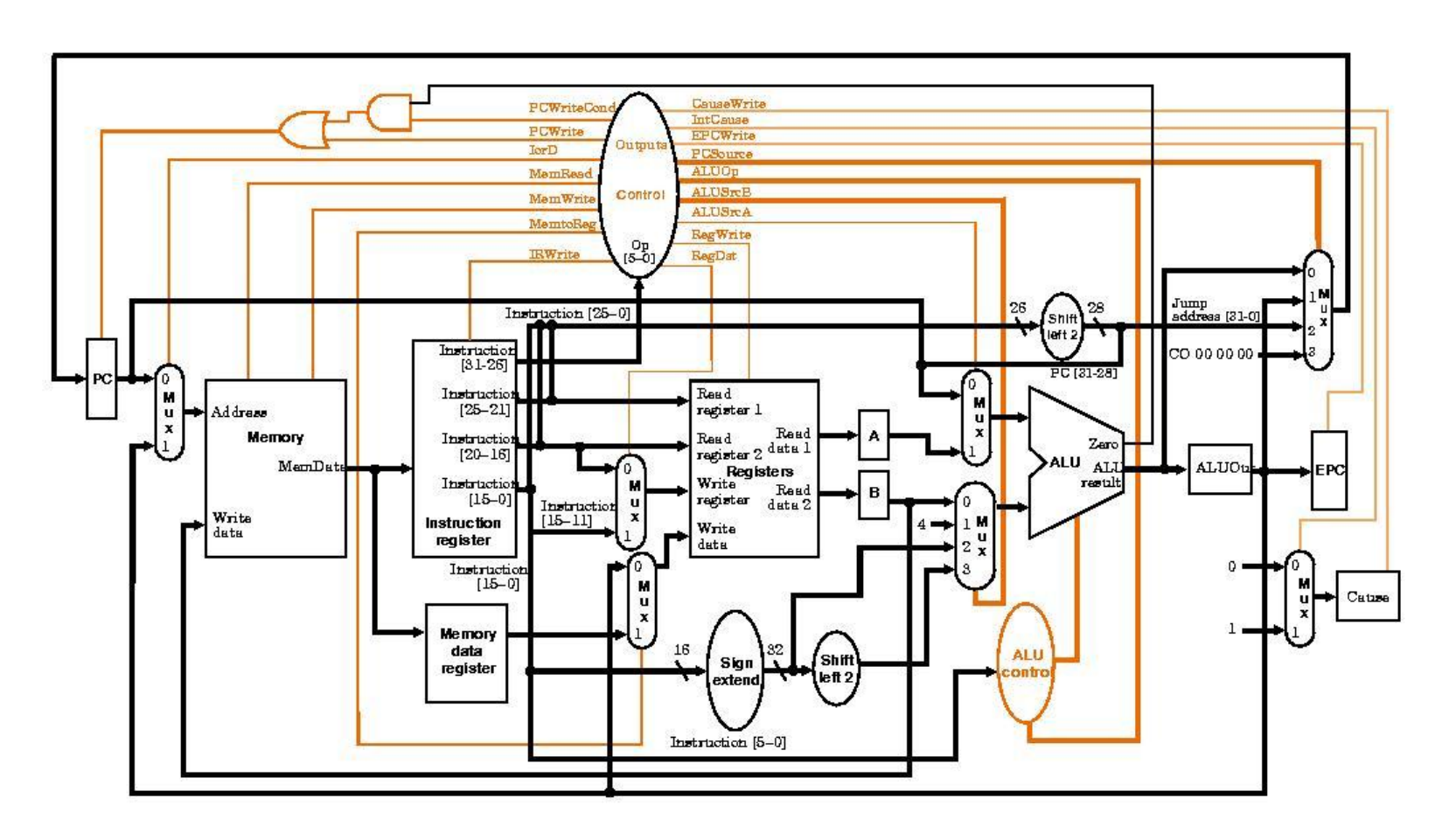
# A SIMPLE CPU: PC, ALU, ACCUMULATOR, IR, MEMORY, BUSES, CONTROL LOGIC



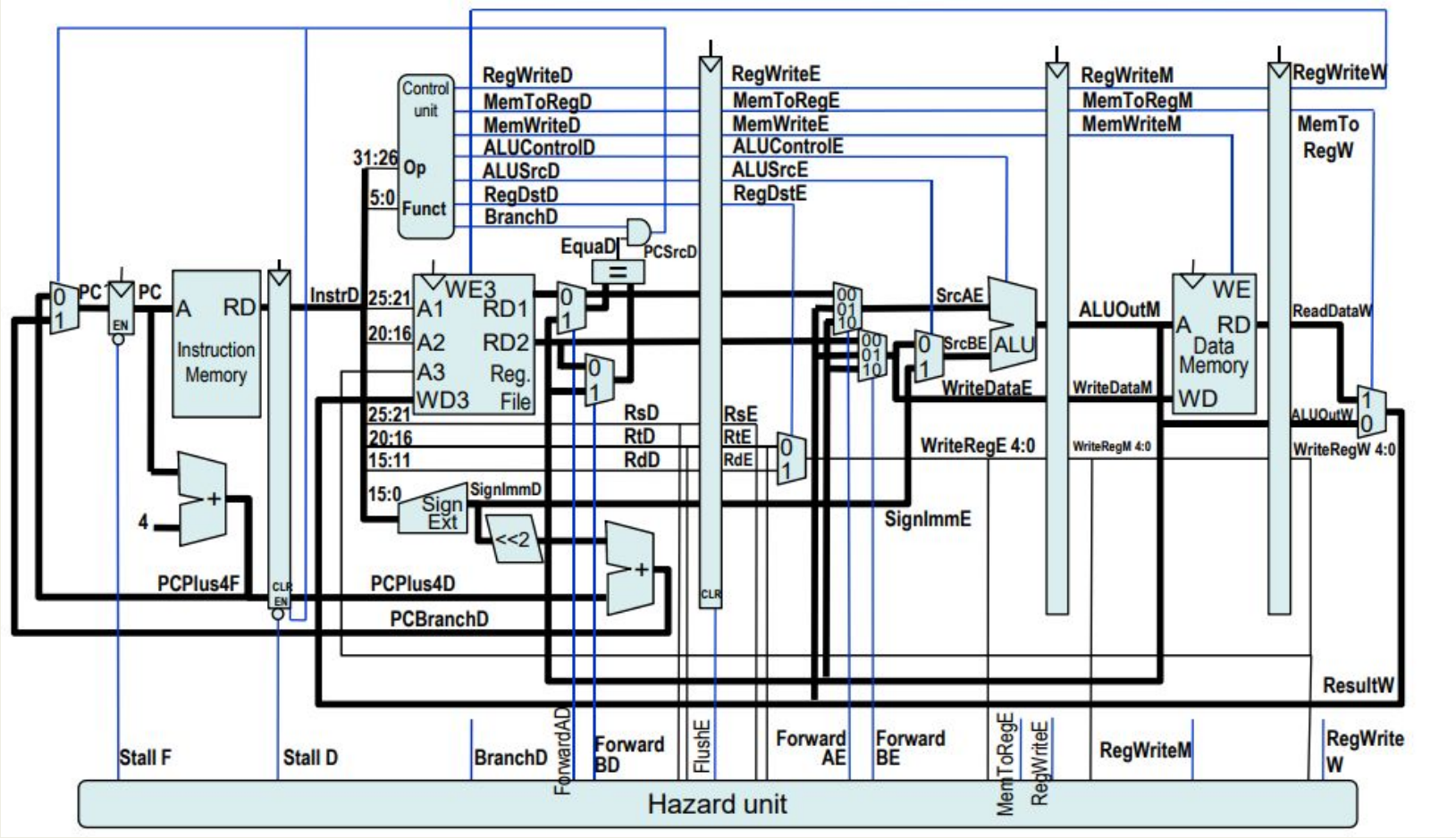
# INSTRUCTIONS AND PROGRAMMING

- Instructions are the most basic actions the processor can process:
  - ADD AX, BX # Add value AX to BX and store in AX
  - CMP AX, 5 #Compare value in AX to 5
  - JE 16 #Jump ahead 16 bytes if comparison was equal
- Instruction Set Architecture (ISA) defines all the instructions of an architecture (e.g. ARM, X86)
- High level programming languages (C, C++, Java) allow many processor instructions to be written simply:
- – if (A + B = 5)
  - then… #Jump if sum of A and B is 5
- Every program (e.g. C) must be converted to the processor instructions (e.g. binaries) of the computer it will be run on

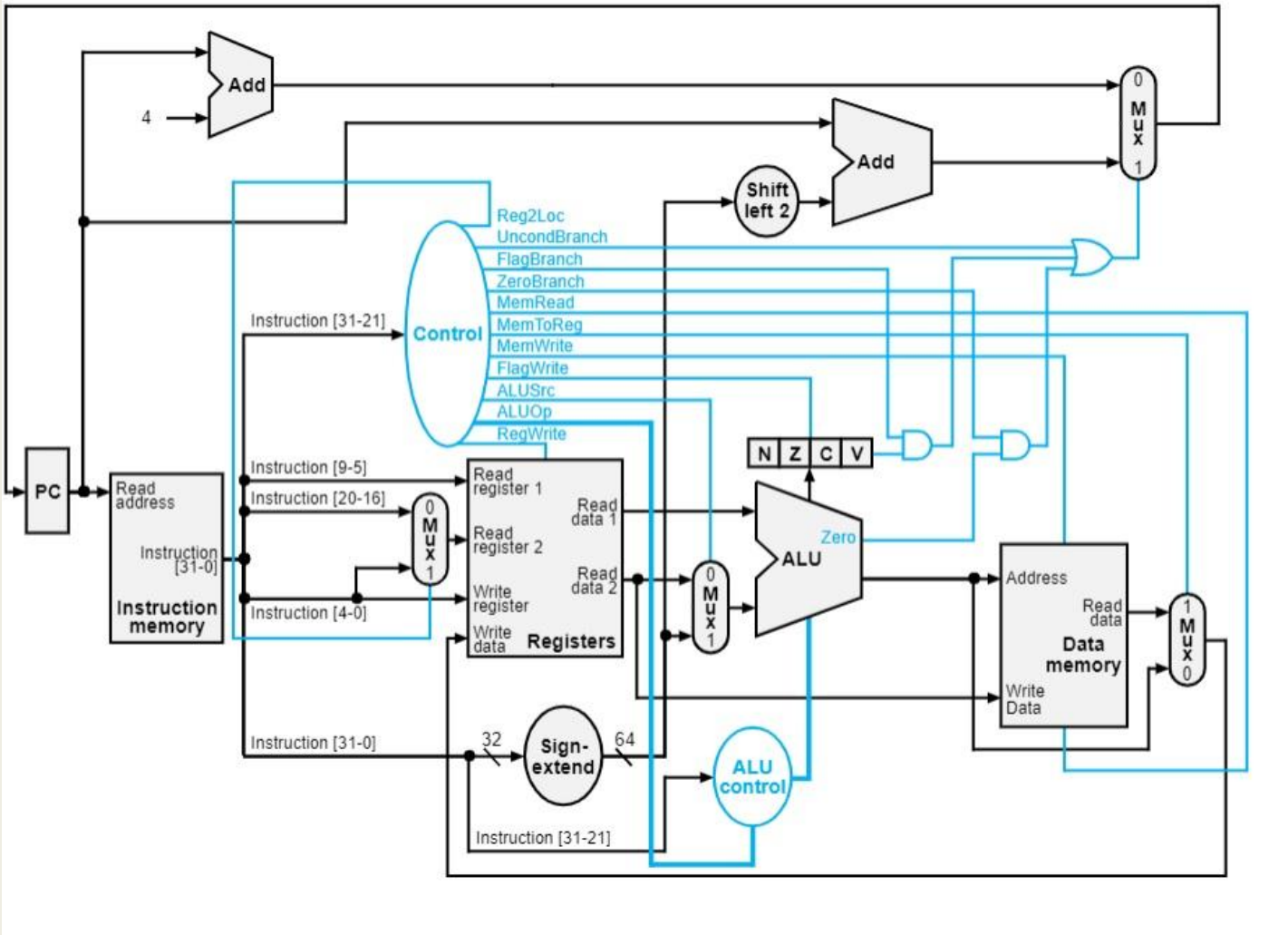
# MIPS architecture



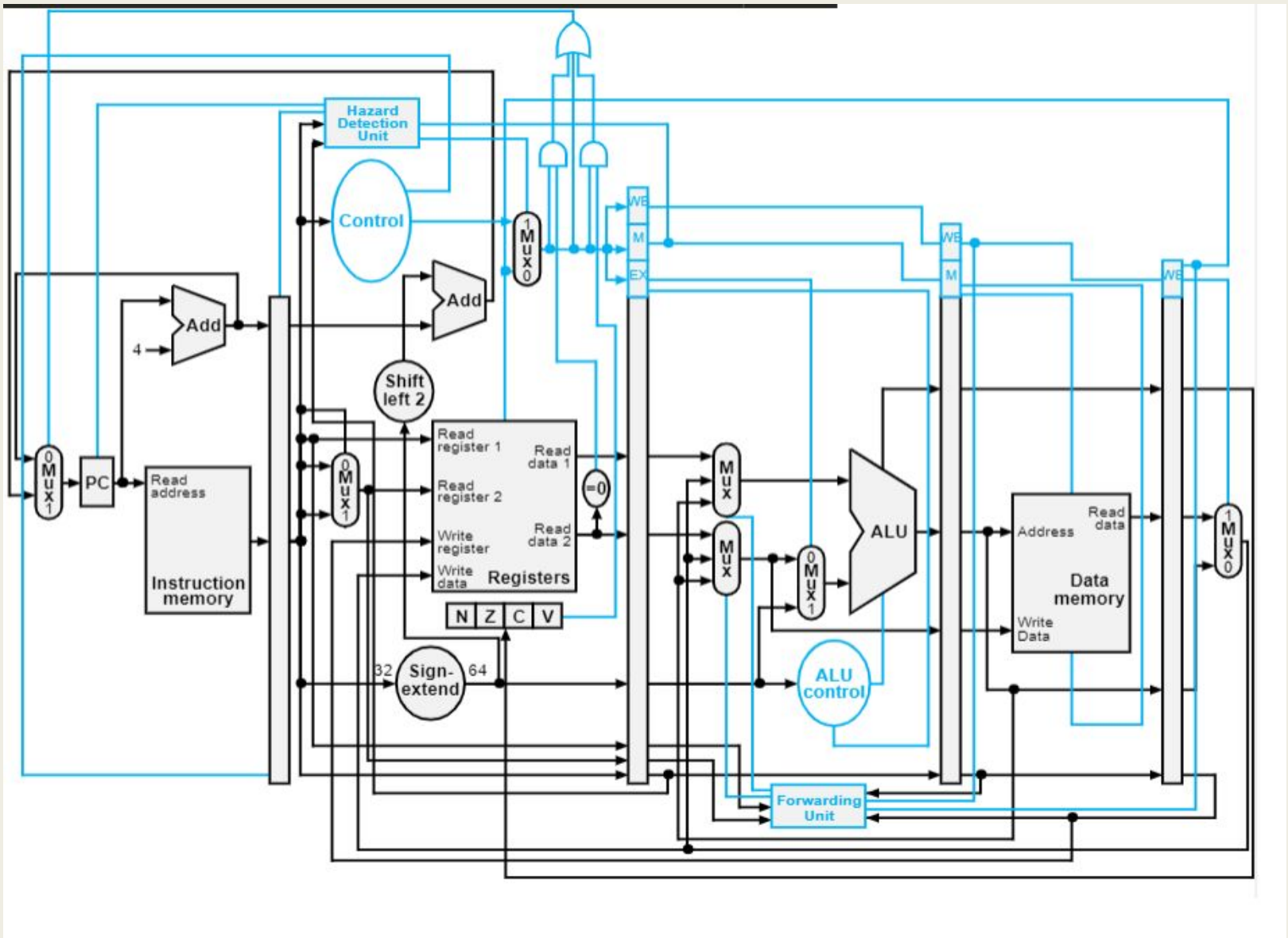
# MIPS architecture with pipeline and hazard control



# ARM architecture



# ARM architecture w Pipeline



# A Instruction Stages Example

- Let's execute the instruction ADD R4,R3,R2
- It is a 32 bit instruction located in Instruction Memory (I-cache)
- This instruction means: add contents of register R2 to the contents of register R3, and store the results into register R4.
- Here's what happens: **Fetch** instruction, **decode**, and read from register addresses 0x00011 (R3) and 0x00010 (R2)
- Connect register R2 to input 1 of the ALU and Connect register R3 to input 2 of the ALU.
- Set controls to ALU to do a 32-bit add operation (**Execute**)
- Store the output of the ALU into register R4 (i.e. connect R4 to output of ALU) **(Write)**
- Increment the program counter and store that value back in the program counter: [PC]=[PC] + 4

# MICROPROCESSOR ARCHITECTURE

- Microprocessor (Computer) architecture is defined by the instructions (ISA) a processor can execute
- • Programs (e.g. C) written for one processor can run on any other processor of the same architecture
- Current architectures include:
  - IA32 also known as x86
  - IA64 also known as x86_54 (Intel, AMD)
  - ARM (Qualcomm, Apple)
  - PowerPC (IBM)
  - MIPS
  - RISC V (Open Source)
  - Xtensa (ESP32)

# Simulator

LEG8 Simulator: ARM University Education Program
https://github.com/arm-university/Graphical-Micro-Architecture-Simulator

Browser Based

Needs to be downloaded

CPUlator: University of Toronto
https://cpulator.01xz.net/doc/

Browser Based

No need to be downloaded

```
@.org    0x1000  // Start at memory location 1000
.data
numbers:
   .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
num_count:
   .word 30
.text  // Code section
.global _start
_start:
@ Initialize the sum to 0
   mov r0, #0
   @ Load the number of elements (30) into r1
   ldr r1, =num_count
   ldr r1, [r1]
   @ Load the address of the numbers array into r2
   ldr r2, =numbers
|
```

```
loop:
   @ Load the current number into r3
   ldr r3, [r2], #4  @ Load the value at the current address and increment the address by 4 bytes
   @ Add the current number to the sum in r0
   add r0, r0, r3
   @ Decrement the counter (r1) by 1
   subs r1, r1, #1
   @ Check if the counter has reached 0 (end of loop)
   bne loop  @ Branch to 'loop' if r1 is not equal to 0
   @ The loop is done; r0 contains the sum
   @ Put your further code here or return from the function if needed
   @ For example, to exit a program in ARM assembly, you can use the following:
   mov r7, r0  @ syscall number for "exit"
// End program
_stop:
B _stop
.end
```

END OF LECTURE#2