# EE 213 Computer Organization and Assembly Language

**Week # 1, Lecture # 2**

**14th Dhu'l-Hijjah, 1439 A.H**

**29th August 2018**

These slides contains materials taken from various sources. I fully acknowledge all copyrights.

# Revision of Topics from Previous Lecture

- Analog transistors as switches. Gates are digital building blocks. Functional modules, like 4 bit adders, are created from gates.
- Processor is a complex digital circuit.
- Computer Organization (also called micro-architecture) is the CPU architecture.
- ISA is a set of operations and related rules & procedures which micro-architecture understands and executes.
- Hardware software boundary is ISA.
- Processor = Micro-architecture + ISA + Caches + other digital logic.
- This means processor is a programmable digital circuit. It accept any sequence of operations as machine code and executes them. It takes input data from and save results to the memory.
- Humans programs in high-level languages, but processor understands machine-code.
- Machine code are instructions which CPU's micro-architecture executes. It is difficult for human to understand.
- HLL code -> Compiler -> Linker -> machine code.
- Assembly code -> Assembler (just like compiler but process assembly code) -> linker -> machine code.
- Assembly Language is human understandable but contains elements which can identify operations and elements specific to a given micro-architecture.
- We learn assembly to understanding working of micro-architecture and not for general purpose programming.

# Today's Topics

- Revision of Previous Lecture's topics
- Computations and computational tools.
- How to design a programmable digital circuit?
  - Brainstorming (5 minutes)
- Block diagram of a digital circuit which execute programs
- Defining a processor/microprocessor/CPU
- Processor = Micro-architecture + ISA
- How to use Processor to execute High-level languages?
- Why knowing Processor's micro-architecture important in CS?
- Future: Big Corps AI Chip Strategy

# Computations

- Computations (why?)
- Doing computations without tools
- Computations with various tools
  - Ancient Computer (Pre 1940)
  - History of Computers (1940 onwards)

  See two presentations in Handout # 1

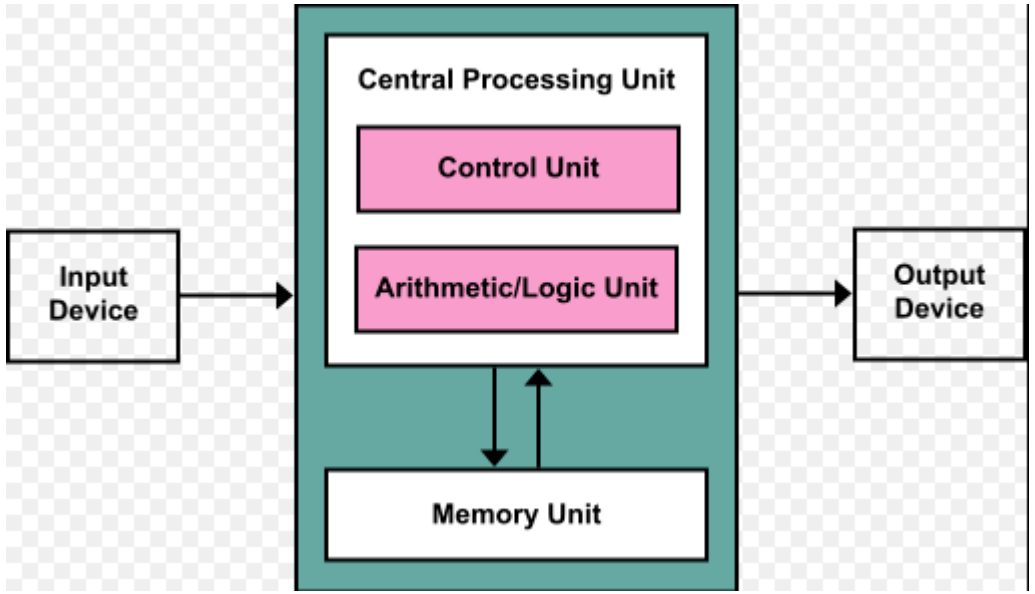
# How to design a programmable digital circuit? (1)

- A digital circuit performs a pre-define function. For example an XOR gate, a 4 bit adder, a 4 bit shift register, a 4 bit counter, BCD to binary decoder.

- How you can design a digital circuit which is programmable? i.e. it performs different operation based on a given program

# Block diagram of a digital circuit which execute programs

- They should read executable code and data from memory (OS load executable code in memory).
- Should be able to read and write to devices like keyboard, computer displays, hard disk, USB devices, network cards, etc.
- Executable code may contains many operations and data items.
- Question: How to process data by executing code?




- A processing unit that contains an arithmetic logic unit and processor registers
- A control unit that contains an instruction register and program counter
- Memory that stores data and instructions
- External mass storage
- Input and output mechanisms[1][2]

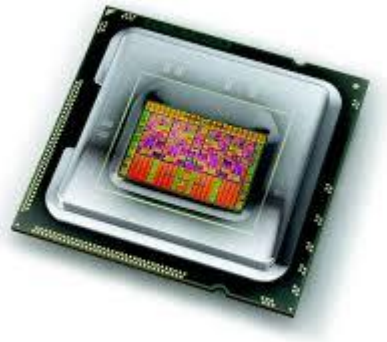
Von Neumann architecture

# How to design a programmable digital circuit? (2)

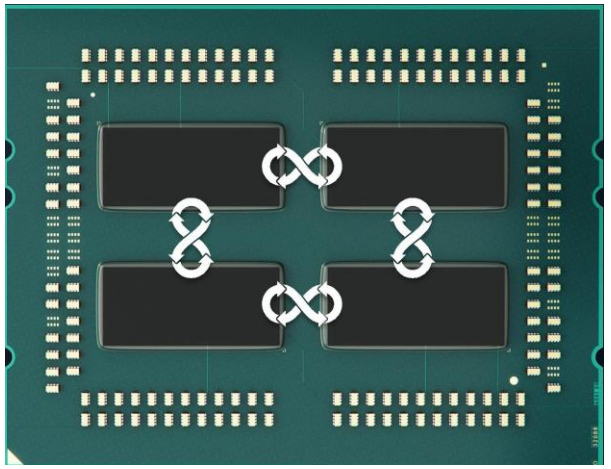
- How you can design a digital circuit which is programmable? i.e. it performs different operation based on a given program.
  - Inputs: interactive inputs
  - Memory: instructions (containing operations) + data (inputs and outputs)
  - CPU (processor): reading instruction from memory + executing them
  - Outputs: write resulting data to devices other than memory

# Defining a processor/microprocessor/CPU

- A microprocessor is a computer processor that incorporates the functions of a central processing unit on a single (or multiple) integrated circuit (IC)
- The microprocessor is a multipurpose, clock driven, register (memory) based, digital-integrated circuit that accepts binary data as input, processes it according to instructions stored in its memory, and provides results as output.
- Microprocessors contain both combinational logic and sequential digital logic.
- Microprocessors operate on numbers and symbols represented in the binary numeral system.



Single-die CPU



Multi-Die CPU

https://www.extremetech.com/computing/251255-amd-takes-fight-intel-new-epyc-server-chips

# Processor = Micro-architecture + ISA

- How to describe all the operations and features of a given micro-architecture?
- Instruction Set Architecture
  - Set of all possible instruction (operations) for a particular processor.
  - Each operation is specified as an instruction and have an associated binary representation (called machine code).
  - Processor ONLY understand these binary bits through signals received on its pin.
  - Compiler or assembler convert HLL or Assembly program into machine code.
- How to design a new processor?
  - Step #1: Decides the operation and functions it has to offer in terms of ISA.
  - Step #2: Hardware team design micro-architecture based on ISA.

# How to use Processor to execute High-level languages?

- We program in High-level languages.
- Processor = Micro-architecture + ISA
- In order to execute program on a particular processor, we need to covert HLL code into processor specific machine code is required.
- How to fill this gap? High-level code vs machine-code
  - Compilers fill this gap by reading high-level language programs and generating low-level language (machine-code) for execution on a processor.
  - Write in machine-code OR manually convert high-level code into machine-code OR writing in some short hand language. (extreme way to understand processor design)
- Processors have extra digital circuitry which does not execute code but provide support for (or speed up) execution.

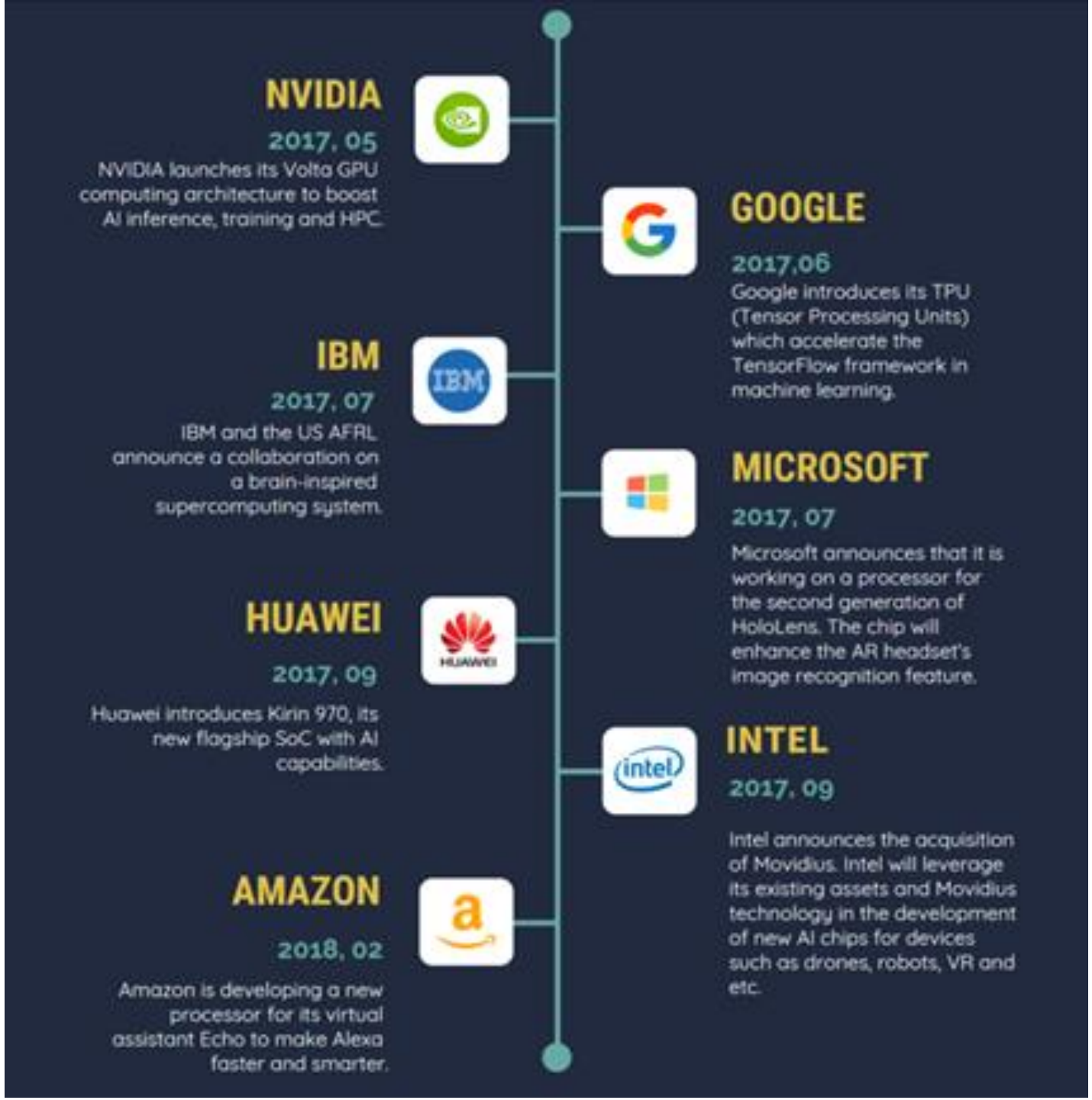
# Why knowing Processor's micro-architecture important in CS?

- Why learn internal working of processors in the age of compilers and hardware virtualization (JVMs, VMs).
- However, the current computational challenges (AI and Big Data) needs:
  - graphics processing unit (GPU),
  - field-programmable gate array (FPGA) and
  - Coarse-Grain Reconfigurable Arrays (CGRAs)
- Large-scale computational problems require computational thinking (expressing solution part) to map the problem into different high-performance hardware architectures.
- Consideration of their internal architecture is important if you want to make use of the off-the-shelf GPUs, FPGA and CGRAs technologies.

# Future: Big Corps AI Chip Strategy

## AI chips for big data and machine learning: GPUs, FPGAs, and hard choices in the cloud and on-premise

How can GPUs and FPGAs help with data-intensive tasks such as operations, analytics, and machine learning, and what are the options?




# Big Corps AI Chips Strategy