M1 Intro + Background

1. Key features of Computer Architecture (condensed)

1.1 Basic Components

ALU / Computing

- Arithmetic logic unit (ALU) performs integer arithmetic operations and logical operations such as add, subtract, multiply, divide, AND, OR, XOR, NOT etc
- Deals with fixed point numbers only

Control Unit

- Control unit generates control signals for data movement and data storage operations, and/or to let the ALU perform the operations specified in the instruction.
- Generated using hardware or microprogram

Memory

I/O

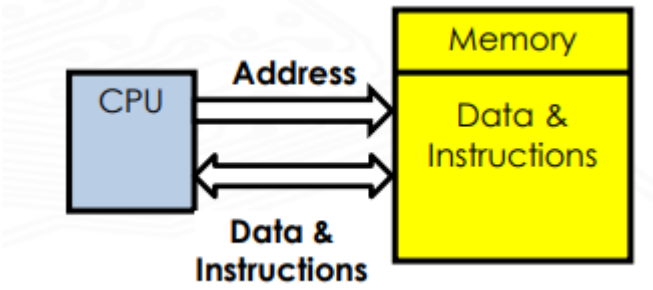
Bus / Communication

CPU = ALU + Storage + Control
Outside CPU = Peripheral + Bus + Storage
Data path is a part of CPU where data is processed / stored / moved through

- Performs all arithmetic and logical operations
- Consists of ALU, registers, on-chip cache and internal buses

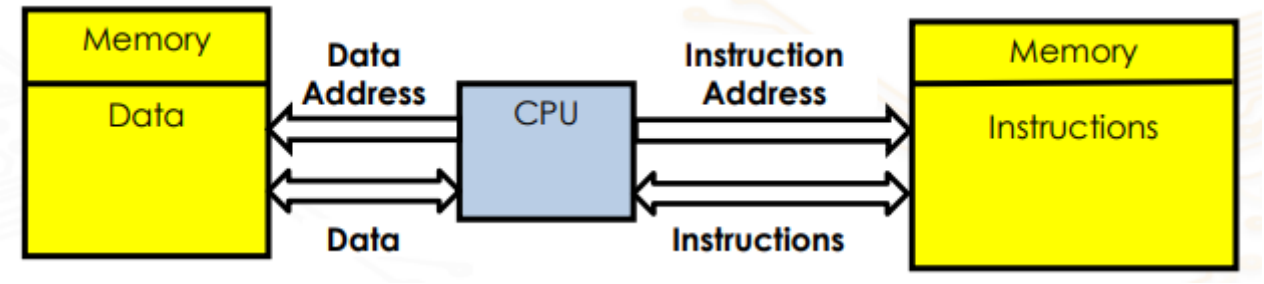
Architectures

- Von Neumann
  - Memory holds **both data and instructions**, transferred through the same bus



  - 1 memory access per clock cycle
- Harvard

- **Separate memory** holds data and instructions, transferred through different buses



- At a snapshot in time, data access corresponding to the previous instruction can be performed while fetching the current instruction

Program Execution

1. Instruction Fetch
   - fetch instruction from memory
2. Instruction Decode
   - decodes fetched instruction, generate control signals and fetch register operand from register file
3. Execute
   - execute ALU operation as specified in the opcode of the instruction
4. Memory Access
   - perform read / write for load / store operations
5. Write back
   - write result back to register file

Design goals and constraints

- Functional Requirement
- Reliability
- Cost
- **Performance**
- **Power Consumption**

2. Performance Metrics, Enhancement Techniques

2.1 Performance

- **Indicated by Execution time** $\rightarrow \text{Performance} = \frac{1}{\text{Execution Time}}$
  - $\text{Execution Time} = T_{end} - T_{start}$

- Execution Time = IC x CPI x T
  - Instruction count (IC)
    - Application/program
    - Instruction set architecture (ISA)
  - Clocks per instruction (CPI)
    - Instruction set architecture (ISA)
    - Datapath design
    - Parallel and pipelined HW design
  - Clock period (T)
    - Semiconductor technology
    - Datapath design and implementation
  - Decrease in one may lead to increase in other two.

2.2 Challenges on Performance Enhancement

- Reduction of clock cycle (T)
  - Power consumption increases
  - Memory operations may take longer than a clock period
    - leads to memory-wall problem (memory becomes slower than CPU, CPU has to wait for data and instructions)
- Reduction of Instruction Count (IC)
  - More complex instructions $\rightarrow$ CPI increase
- Reduction of Cycles per Instruction (CPI)

2.3 Speedup

- $\text{Speedup} = \frac{\text{Perf}_A}{\text{Perf}_B} = \frac{\text{Time}_B}{\text{Time}_A} = \frac{T_{\text{original}}}{T_{\text{enhanced}}}$
  - **remember performance is inversely proportional to time**

If fraction E of the program is enhanced by a factor of S in an enhanced machine, then determine the speedup of the enhanced machine over the machine before enhancement (original machine). What is the maximum speed up for a given E if S can be any value greater than or equal to 1?

Fraction $U = (1 - E)$ of the program is not enhanced.
if T is the execution time of the program in the unenhanced (original) machine,

Time required for the execution of unenhanced fraction = $T \times (1 - E)$
Time required for the execution of enhanced fraction = $[T \times E]/S$

Total execution time in the enhanced machine

$$T' = [T \times (1 - E)] + [T \times E]/S$$

$T' = T \times \left[(1 - E) + \frac{E}{S}\right] = T \times [1 - E(S - 1)/S]$ *(check: if $S = 1$ then $T' = T$).*

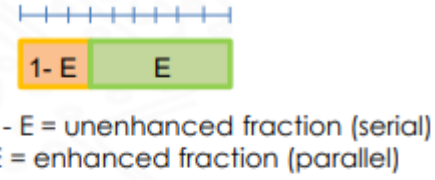
Speed up = $\frac{T}{T'} = \frac{T}{T} / \left[(1 - E) + \frac{E}{S}\right] = \frac{1}{[(1-E)+E/S]}$

If S is very large (i.e., $S \to \infty$) then $[(S - 1)/S] \to 1$, and $T' \to T \times (1 - E)$
So the maximum speedup = $T/T' = T/[T \times (1 - E)] = 1/(1 - E)$
(Note that (1- E) is the unenhanced or sequential fraction of the program.)

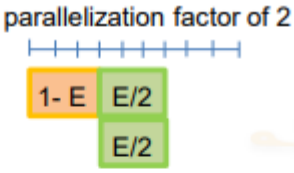
If (1 - E)= (1/10) then speedup = 10, i.e., if 1/10th of the program is not-enhanced then, maximum achievable speedup is 10.
If (1 - E)= (1/5) then speedup = 5, i.e., if (1/5)th of the program is not-enhanced then, maximum achievable speedup is 5.
If (1 - E)= 1/2; 1/3 or 1/4 ????

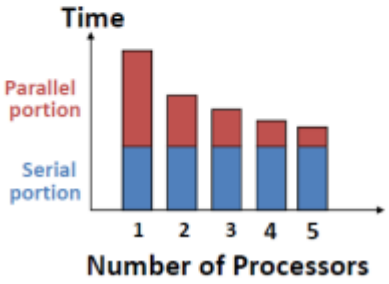
2.4 Amdahl's Law

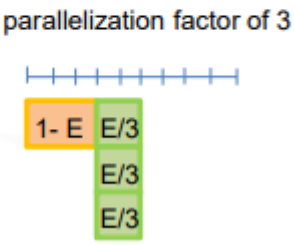
- Speedup via parallelism is limited by *that component* (sequential / serial component) of an application which cannot be enhanced



1- E = unenhanced fraction (serial)
E = enhanced fraction (parallel)



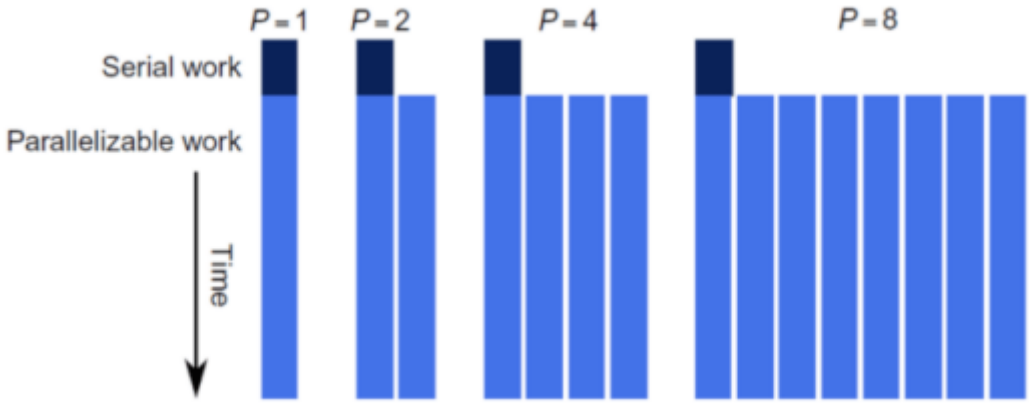




- $\text{Speedup}_{\max} = \frac{1}{1-E}$

2.5 Gustafson's Law

- A program can be ran in parallel with other programs to increase workload given the same time to run just one program



  - each blue bar represents another program running in parallel with the initial program
- $\text{Speedup}_{\max} = \frac{T_s + n \times T_p}{T_s + T_n}$
  - n = number of programs running in parallel

2.6 Metrics

- Instructions per Second (IPS) = $\frac{\text{Instruction Count}}{\text{Execution Time}}$
- Million IPS = $\frac{\text{IPS}}{10^6}$
- Relative MIPS = $\frac{T_{\text{ref}}}{T_{\text{machine to be rated}}} \times \text{MIPS}_{\text{ref}}$

  T_{ref} : Execution time of reference machine
  $T_{machine\_to\_be\_rated}$: Execution time machine to be rated
  - MIPS varies with
    - the ISA (i.e., the complexity of instructions)
    - the choice of instruction mix (program)
  - Higher MIPS does not guarantee better performance (instruction complexity)
  - Relative MIPS is useful to rate evolving designs of the same computer
- $\text{FLOPS} = \frac{\text{Number of Floating Point Operations}}{\text{Execution Time}}$

3. Power Dissipation, Power Metrics, Low-Power Design Techniques

3.1 Power Dissipation

- **Dynamic Power** $P_{dyn} = ACV^2 f$
  - dissipated when processor executes instructions

- increases with the operating voltage (V) and clock frequency (f)

C: total load capacitance in the circuit
f: clock frequency
Reduction of frequency reduces P_{dyn}, but can degrade performance
V : operating voltage (also called V_{dd})
Reduction of voltage reduces power consumption significantly
A: switching activity factor: the fraction of transistors switch during a clock cycle (in average).
Can be reduced by turning-off the unused resources/components

- **Static / Leakage Power** $P_{st} = VI_{leak}$
  - dissipated whenever the system is powered-on, even if no computation is done
  - increases with temperature of the processor
- $\text{Total Power Consumption} = ACV^2f + VI_{leak}$
  - reduce V = reduce power consumption
- Maximum operating frequency $f_{max} \propto [V - V_{th}]^2/V$
  - V_{th} is called the threshold voltage, the gate-source voltage at which the transistor just starts conducting
  - if V_{th} is small compared to V, the maximum usable frequency $f_{max} \propto V$
- **Power** Consumption **can** be reduced by reducing only frequency
- **Energy** Consumption **cannot** be reduced by reducing only frequency

4. Key points from LAMS

- If both processors P1 and P2 gives the same performance (given by IC * CPI * Time) , the one with lower frequency (higher clock period) is preferred since it consumes less power
- If a task has more subtasks that can be executed by independent processors, it is easier to do parallelisation
- Pipelining is the process of accumulating instruction from the processor through a pipeline (i.e. instructions are executed in a given order)