19-09-18

Comp Arch 3 - Intro

raylynch.000webhostapp.com
User: ft2814
Pass: paul2814

Classify Computers:

We Want ↓

1. Personal Computers → Response time (User Perspective)
2. Servers (Web, File) → Throughput (transactions/sec)
3. Supercomputers → Throughput
4. Embedded → Reliability

(1) How do we minimise response time?

Topics:
- i. Internal CPU Design
- ii. Memory Design: Caches & DRAM
- iii. Parallel Processing - Multicore
- iv. Storage disks, etc.

Computer Performance Between Computers:

Performance M/C $x = n \cdot$ M/C y

Take a program: measure execution time of x & y

Higher Performance → Shorter execution time

Performance $x = \frac{1}{\text{Execution } t}$ → $\frac{\text{Perf } x}{\text{Perf } y} = \frac{\text{Exec } y}{\text{Exec } x} = n$

Prog takes: 10s to run on A
15s on B

$\frac{15_B}{10_A} = 1.5$ A is 1.5 times faster than B

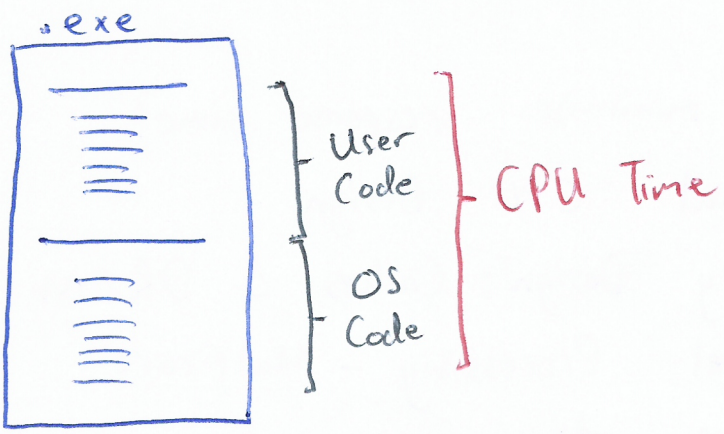
Measure execution time of program?

- Governed by current activity on multitasking mcu
- Execution time governed by: CPU Exec t, I/O, other

CPU Time:

Ray.c

```
main
{
  int, x, y;
  x=y;
  source
}
```



→ Ideally would like to measure ~~the~~ execution time of
- User code (Usually only able to measure this)
- OS code

→ User code is easy to access, O/P of PC - MCU ~~code~~ code instructions (assembly)

→ Have figures for how long every instruction takes

→ Program composed of MCU code instructions

→ Exec Time = CPU Time Executing Instructions

Comp Arch 3 - Intro

19-09-18

→ Instruction Execution measured by no. of clock cycles / instructions

→ Execution time of program = no. of clock cycles x T_{clock}

* ~~A~~verage no. of cycles/inst can vary instruction to instruction (CPI)

= no. of prog inst x Avg(cycles / instr) * x ~~T_{clc}~~ x T_{clock}

Analyse Computer Program:

Execution Time of program: No. of Instr x CPI x T_{clock}

$$\text{Exec Time} = \frac{\text{Instr Count} \times \text{CPI}}{\text{~~I_{clock}~~ Clk Rate}}$$

Example:

	Cycle Time	CPI
MCU (A)	250ps	2
MCU (B)	500ps	1.2

~~wh~~ Which one is faster? Assuming they have identical Instruction Set of Architecture (ISA)

Exec (A) = ISA x 2 x 250ps = 500 x ~~I~~ ISA

Exec (B) = ISA x 1.2 x 500ps = 600 x ISA

$\frac{\text{Exec (B)}}{\text{Exec (A)}} = \frac{600}{500} = 1.2$ (A) is 20% faster than (B)

∴ ~~Performance depends upon~~

Performance depends on: Algorithm → ISA + CPI

Language & Compler → ISA + CPI

ISA = N, CPI, T_{clock}

Amdalil's Law:

→ Applies to making changes to hardware to improve overall performance

→ Improve Multiplier Area of CPU: Works faster but don't change other parts of CPU

$$T_{overall\ pof} = \frac{T_{multiplier}}{Improvement} + T_{unmodified}$$

In a sample program multipliers take 80s for 100s of program

How must we improve multiplication perf to get the MCU to run 5 times faster?

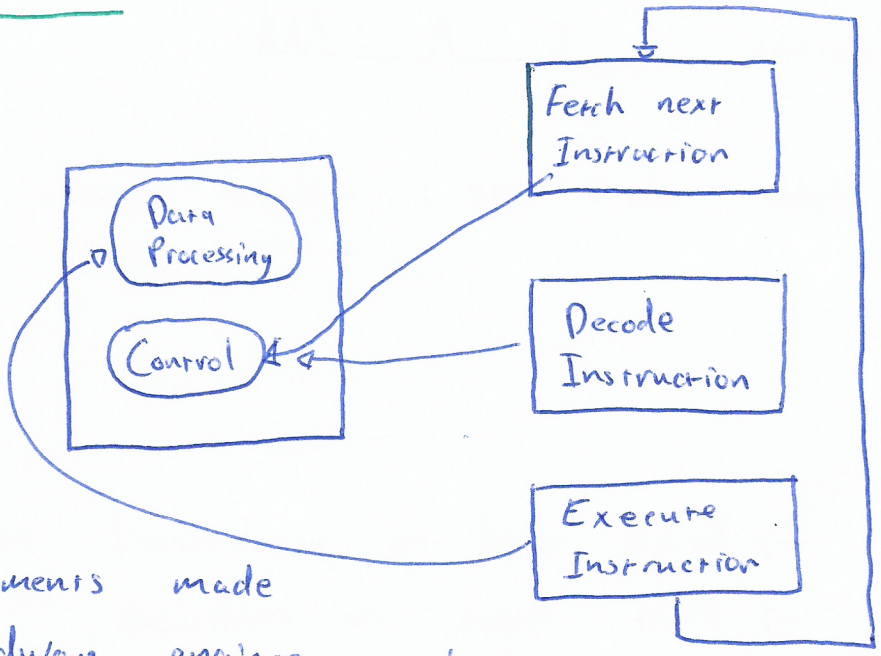
Moore's Low:

Density of transistors will double every 18 months ≈ CPU power doubles every 18 month

19-09-18

Comp Arch 3 - Intro

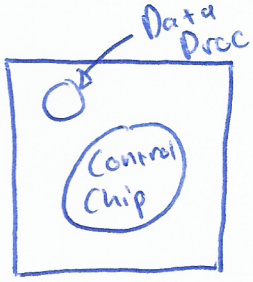
CPU Breakdown:



→ Improvements made by Hardware engineers w/ extra transistors → no work for software

Instruction Level Parallelisation (ILP):

→ Execute multiple instructions in parallel at same time

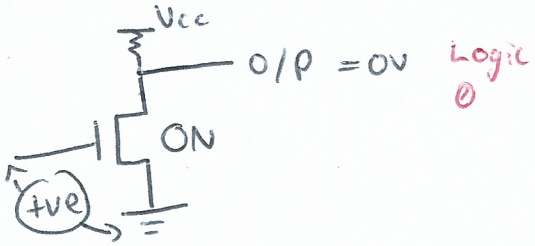


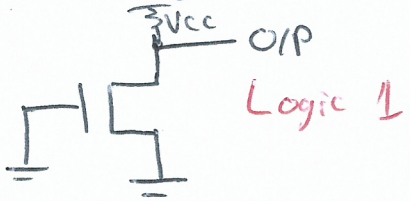
ILP improvements slowed down

∴ control size >> data processing

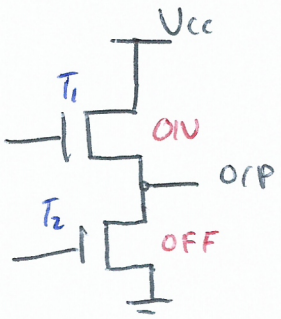
→ power consumption becomes issue

→ As transistors are reduced in size





↳ Transistors smaller: problem switching fully off



Voltage o/p 

Current 

→ Higher clock rate : • Higher current
• More power

- ILP Wall
- Power Wall
- Memory Wall
  ↳ Brick wall to traditional unicore CPUs
  → Must move to multicore

→ Use simpler internal design, but use CPUs in parallel

→ Up to software engineers to ~~reduce~~ design programs for multicore platform - multithreaded program