

## L2: Background Parts 1 & 2 – Historical Context & Performance

## Outline

- Review Introduction (another perspective)
- Background 1: Historical Context
- Background 2: Performance

COMP3211/9211

2003 S2 L2 P2

## Review of Introduction

## What is “Computer Architecture”?

Computer Architecture =  
Instruction Set Architecture +  
Machine Organization

COMP3211/9211

[Patterson]

2003 S2 L2 P4

## Example Instruction Set Architectures (ISAs)

- |                 |  |      |
|-----------------|--|------|
| • Digital Alpha | (v1, v3)                                       | 1992 |
| • HP PA-RISC    | (v1.1, v2.0)                                   | 1986 |
| • Sun Sparc     | (v8, v9)                                       | 1987 |
| • SGI MIPS      | (MIPS I, II, III, IV, V)                       | 1986 |
| • Intel         | (8086, 80286, 80386, 80486, Pentium, MMX, ...) | 1978 |

COMP3211/9211

[Patterson]

2003 S2 L2 P5

## MIPS R3000 ISA (Summary)

- Main instruction categories
  - Load/Store
  - Computational
  - Jump and Branch
  - Floating Point
    - coprocessor
  - Memory Management

Registers

R0 - R31

PC

3 Instruction Formats: all 32 bits wide

OP	rs	rt	rd	sa	funct
----	----	----	----	----	-------

OP	rs	rt	immediate
----	----	----	-----------

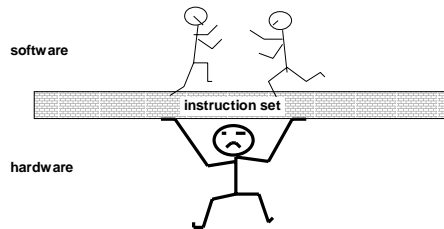
OP	jump target
----	-------------

COMP3211/9211

[Patterson]

2003 S2 L2 P6

## The Instruction Set: a Critical Interface

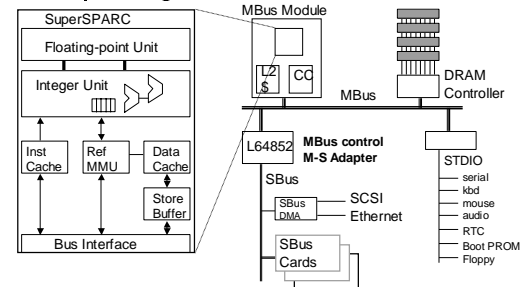


COMP3211/9211

[Patterson]

2003 S2 L2 P7

## Example Organization



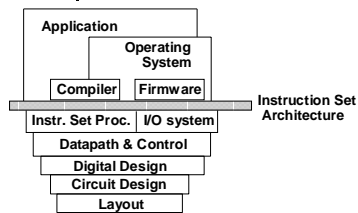
TI SuperSPARC™ TMS390Z50 in Sun SPARCstation20

COMP3211/9211

[Patterson]

2003 S2 L2 P8

## What is “Computer Architecture”?



- Coordination of many levels of abstraction
- Under a rapidly changing set of forces
- Design, Measurement, and Evaluation

COMP3211/9211

[Patterson]

2003 S2 L2 P9

## Course Content

### Computer Architecture and Engineering

Instruction Set Design	Computer Organization
Interfaces	Hardware Components
Compiler/System View	Logic Designer's View
-“Building Architect”	-“Construction Engineer”

COMP3211/9211

[Patterson]

2003 S2 L2 P10

## Background 1: Historical Context

## ENIAC - background

- Electronic Numerical Integrator And Computer
- Eckert and Mauchly
- University of Pennsylvania
- Trajectory tables for weapons
- Started 1943
- Finished 1946
  - Too late for war effort
  - But early enough to be used in the design of the Hydrogen bomb
- Used until 1955
- But was it really the first?

COMP3211/9211

[Stallings]

2003 S2 L2 P12

## ENIAC - details

- Decimal (not binary)
- 20 accumulators of 10 digits
- Programmed manually by setting switches and plugging and unplugging cables
- 18,000 vacuum tubes
- 30 tons
- 15,000 square feet
- 140 kW power consumption
- 5,000 additions per second

COMP3211/9211

[Stallings]

2003 S2 L2 P13



Reaching a hot wire man checking among ENIAC's 18,000 switches

## von Neumann/Turing

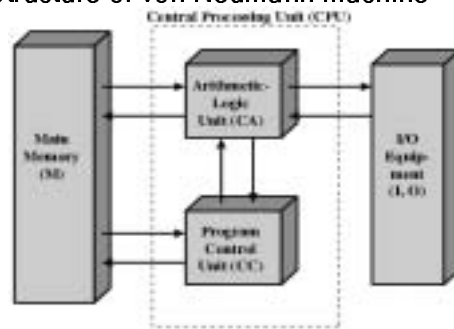
- Stored Program concept
- Main memory storing programs and data
- ALU operating on binary data
- Control unit interpreting instructions from memory and executing
- Input and output equipment operated by control unit
- Princeton Institute for Advanced Studies
  - IAS begun 1946 & completed 1952
  - Became prototype for all subsequent "general purpose" computers

COMP3211/9211

[Stallings]

2003 S2 L2 P15

## Structure of von Neumann machine



COMP3211/9211

[Stallings]

2003 S2 L2 P16

## IAS - details

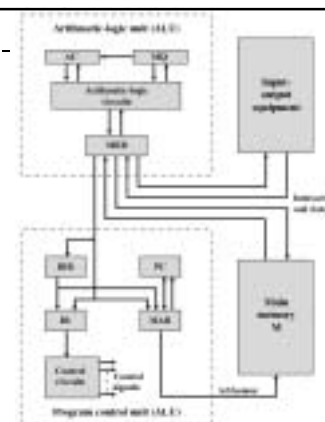
- 1000 x 40 bit words
  - Binary number
  - 2 x 20 bit instructions - 8-bit opcode; 12-bit address/operand
- Set of registers (storage in CPU)
  - Memory Buffer Register
  - Memory Address Register
  - Instruction Register
  - Instruction Buffer Register - stores right hand instruction
  - Program Counter - pointer to next instruction pair
  - Accumulator
  - Multiplier Quotient - least significant word of mult/div

COMP3211/9211

[Stallings]

2003 S2 L2 P17

## Structure of IAS - detail



COMP3211/9211

## Commercial Computers

- 1947 - Eckert-Mauchly Computer Corporation
- UNIVAC I (Universal Automatic Computer)
  - Scientific & commercial applications e.g., matrix algebra, statistical problems, premium billings for life insurance companies, and logistical problems
- US Bureau of Census 1950 calculations
- Became part of Sperry-Rand Corporation
- Late 1950s - UNIVAC II
  - Faster
  - More memory

COMP3211/9211

[Stallings]

2003 S2 L2 P19



## IBM

- Punched-card processing equipment
- 1953 - the 701
  - IBM's first stored program computer
  - Scientific calculations
- 1955 - the 702
  - Business applications
- Lead to 700/7000 series

COMP3211/9211

[Stallings]

2003 S2 L2 P21

## Transistors – 2<sup>nd</sup> Generation

- Replaced vacuum tubes
- Smaller
- Cheaper
- Less heat dissipation
- Solid State device
- Made from Silicon (Sand)
- Invented 1947 at Bell Labs
- William Shockley et al.
- Completely transistorized computers by late '50s

COMP3211/9211

[Stallings]

2003 S2 L2 P22

## Transistor Based Computers

- Second generation machines
- NCR & RCA produced small transistor machines
- IBM 7000 followed
- DEC - 1957
  - Produced PDP-1

COMP3211/9211

[Stallings]

2003 S2 L2 P23

## Microelectronics

- Literally - "small electronics"
- The computers gates, memory cells, and interconnections can be manufactured on a semiconductor, e.g. silicon wafer
- Start of integrated circuit design – 3<sup>rd</sup> generation

COMP3211/9211

[Stallings]

2003 S2 L2 P24

## Generations of Computer

1. Vacuum tube: 1946-1957
2. Transistor: 1958-1964
3. Integrated Circuit (IC): 1965+
  - Small scale integration (SSI): 1965 on
    - Up to 100 devices on a chip
  - Medium scale integration (MSI): to 1971
    - 100-3,000 devices on a chip
  - Large scale integration (LSI): 1971-1977
    - 3,000 - 100,000 devices on a chip
  - Very large scale integration (VLSI): 1978 to date
    - 100,000 - 100,000,000 devices on a chip
  - Ultra large scale integration (ULSI): Now +
    - Over 100,000,000 devices on a chip
4. ??? (Choose from Quantum, Nanotech, Biomolecular,...)

COMP3211/9211

2003 S2 L2 P25

## Moore's Law

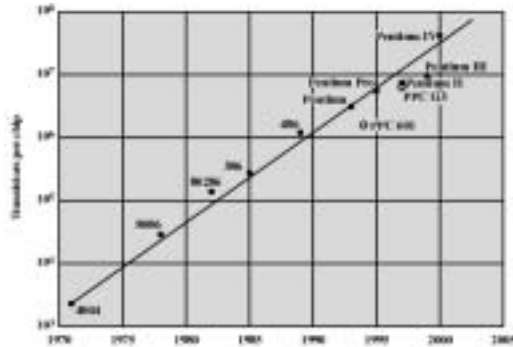
- Relates to the increased density of components on chip
- Discovered by Gordon Moore (cofounder of Intel)
- Number of transistors on a chip will double every year
- Since 1970's development has slowed a little
  - Number of transistors doubles every 18 months
- Cost of a chip has remained almost unchanged
- Higher packing density means shorter electrical paths, giving higher performance
- Fewer interconnections increases reliability
- Reduced power and cooling requirements
- Smaller size gives increased flexibility

COMP3211/9211

[Stallings]

2003 S2 L2 P26

## Growth in CPU Transistor Count



## IBM 360 series

- 1964 (birth of the modern era of computer design?)
- Replaced (& not compatible with) 7000 series
- First planned "family" of computers
  - Similar or identical instruction sets
  - Similar or identical O/S
  - Increasing speed
  - Increasing number of I/O ports (i.e. more terminals)
  - Increased memory size
  - Increased cost
- Multiplexed switch structure

COMP3211/9211

[Stallings]

2003 S2 L2 P28

## DEC PDP-8

- Also 1964!
- First minicomputer (after miniskirt!)
- Did not need air conditioned room
- Small enough to sit on a lab bench
- \$16,000
  - \$100k+ for IBM 360
- Leant itself to embedded applications & resale by third parties
- BUS STRUCTURE

COMP3211/9211

[Stallings]

2003 S2 L2 P29

## PDP-8

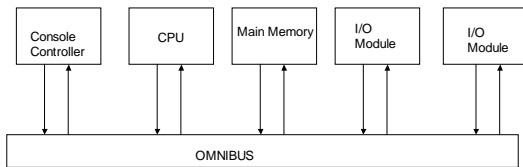


COMP3211/9211

2003 S2 L2 P30

## DEC - PDP-8 Bus Structure

- 96 signal paths for control, address, and data arbitrated by the CPU



COMP3211/9211

[Stallings]

2003 S2 L2 P31

## Semiconductor Memory

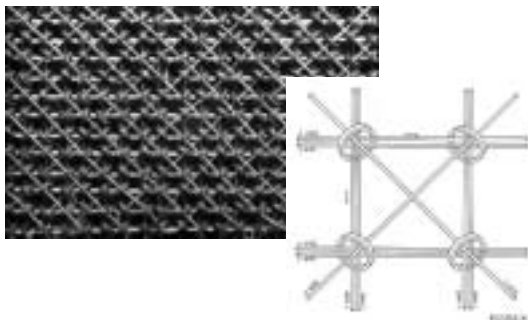
- 1970
- Fairchild
- Size of a single core
  - i.e. 1 bit of magnetic core storage
- Holds 256 bits
- Non-destructive read
- Much faster than core
- Initially more expensive than core
- Capacity approximately doubles each year
- Became cheaper than core in '74

COMP3211/9211

[Stallings]

2003 S2 L2 P32

## Core memory



COMP3211/9211

2003 S2 L2 P33

## Intel

- 1971 - 4004
  - First microprocessor
  - All CPU components on a single chip
  - 4 bit
- Followed in 1972 by 8008
  - 8 bit
  - Both designed for specific applications
- 1974 - 8080
  - Intel's first general purpose microprocessor
- 1979 - 8086
  - Ancestor of Pentium

COMP3211/9211

[Stallings]

2003 S2 L2 P34

## Technology $\Rightarrow$ dramatic change

- Processor
  - Logic capacity: about 30% per year
  - Clock rate: about 20% per year
- Memory
  - DRAM capacity: about 60% per year (4x every 3 years)
  - Memory speed: about 10% per year
  - Cost per bit: about 25% per year
- Disk
  - Capacity: about 60% per year

COMP3211/9211

[Patterson]

2003 S2 L2 P35

## Performance Mismatch

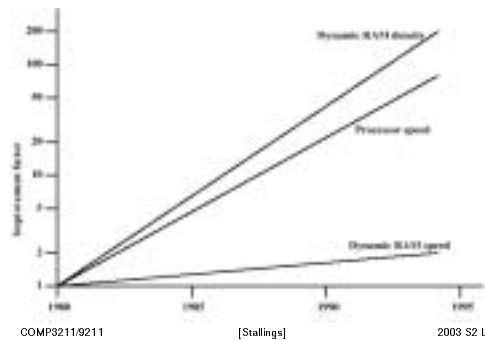
- Processor speed increased
- Memory capacity increased
- Memory speed lags behind processor speed

COMP3211/9211

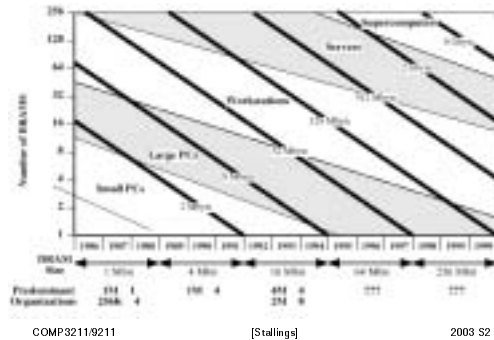
[Stallings]

2003 S2 L2 P36

## DRAM and Processor Characteristics



## Trends in DRAM use



## Solutions

- Increase number of bits retrieved at one time
  - Make DRAM "wider" rather than "deeper"
- Reduce frequency of memory access
- Change DRAM interface
  - Cache
  - More complex cache and cache on chip
- Increase interconnection bandwidth
  - High speed buses
  - Hierarchy of buses

COMP3211/9211 [Stallings] 2003 S2 L2 P39

## Speeding it up

- Pipelining
- On board cache
- On board L1 & L2 cache
- Branch prediction
- Data flow analysis
- Speculative execution

COMP3211/9211 [Stallings] 2003 S2 L2 P40

## Background 2: Performance

## How do we quantify improvements? How do we compare performance?

- Two principle measures:
  1. Response time/execution time/elapsed time
    - How long does it take a job to run?
    - How long does it take to execute a task?
    - How long does it take to return of a query?
  2. Throughput
    - How many jobs can be run at once?
    - What is the average execution rate?
    - How much work is done?
- Examples:
  - Upgrading a machine with a new processor
  - Adding a new machine to the lab

COMP3211/9211 [Gao] 2003 S2 L2 P42

## Response time /execution time

- Elapsed time = finish time - start time
  - Counts everything (disk and memory access, I/O etc)
  - Not so good for purpose of comparison
- CPU time
  - Doesn't count I/O time or time spent running other programs by the CPU
  - Can be broken up into system time, and user time
- Our focus: user CPU time
  - Time spent executing the lines of code that are "in" our program

COMP3211/9211

[Gui]

2003 S2 L2 P43

## Unix time command

- `% time find . - name myfile`  
`90.7u 12.9s 2:39 65%`
  - 90.7u --- user CPU time in seconds
  - 12.9s --- system CPU time in seconds
  - 2:39 --- response time
  - $(90.7 + 12.9)/(2*60+39) = 65\%$
  - 65% of the elapsed time was used by the CPU executing the program

COMP3211/9211

[Gui]

2003 S2 L2 P44

## What is "performance"?

- Performance is in units of things executed per second
  - Higher values intuitively better
- If we are primarily concerned with response time, the performance of machine X is
$$performance(X) = 1/execution\_time(X)$$
- "X is n times faster than Y" means
  - $performance(X)/performance(Y) = n$
  - $execution\_time(Y)/execution\_time(X) = n$

COMP3211/9211

[Gui]

2003 S2 L2 P45

## Aspects of CPU performance

- What are the factors that determine CPU performance?

COMP3211/9211

2003 S2 L2 P46

## Aspects of CPU performance

- Evaluated based on the CPU time of a certain program

$$cpu\_time = \frac{seconds}{program} = \frac{instructions}{program} \times \frac{cycles}{instruction} \times \frac{seconds}{cycle}$$

- CPU time
  - Time in seconds the CPU spends executing a program
  - Comprised of three key components
    - instruction count = instructions/program
    - CPI = cycles/instruction
    - clock period (or cycle time) = seconds/cycle
      - clock frequency =  $1/(\text{clock period})$

COMP3211/9211

[Gui]

2003 S2 L2 P47

## Aspects of CPU performance

- The various levels of computer design affect each aspect differently

Design level	Instr. count	CPI	Clock period
program	X		
compiler	X	X	
instr. set	X	X	
organization		X	X
technology			X

COMP3211/9211

[Gui]

2003 S2 L2 P48



## CPI - average cycles per instruction

$$cpu\_time = clock\_period * \sum_{i=1}^n CPI_i * I_i$$

Where  $n$  = the number of instructions in the ISA

$I_i$  = instruction count for instruction  $i$

$CPI_i$  = number of clock cycles for instruction  $i$

$$\begin{aligned} CPI &= (cpu\_time * clock\_frequency) / instr\_count \\ &= (cpu\_time / clock\_period) / instr\_count \\ &= \sum_{i=1}^n CPI_i * \frac{I_i}{instr\_count} \end{aligned}$$

COMP3211/9211

[Gur]

2003 S2 L2 P49

## Example: Comparing code segments

- A compiler writer must decide between two code sequences for a machine given following facts:

Instruction Class CPI for this inst. class

A	1
B	2
C	3

- The two code sequences require the following instruction counts:

Code sequence	A	B	C
1	2	1	2
2	4	1	1

COMP3211/9211

2003 S2 L2 P50

## Example continued

- Which code sequence executes the most instructions?

Code sequence	A	B	C
1	2	1	2
2	4	1	1

COMP3211/9211

2003 S2 L2 P51

## Example continued

- Which code sequence is faster?

Instruction Class CPI for this inst. class

A	1
B	2
C	3

Code sequence	A	B	C
1	2	1	2
2	4	1	1

- CPU clock cycles<sub>1</sub> = 2x1 + 1x2 + 2x3 = 10 cycles
- CPU clock cycles<sub>2</sub> = 4x1 + 1x2 + 1x3 = 9 cycles

COMP3211/9211

2003 S2 L2 P52

## Example continued

- What is the CPI for each sequence?
- $CPI = (CPU \text{ clock cycles}) / (Instruction \text{ count})$
- $CPI_1 = 10/5 = 2$
- $CPI_2 = 9/6 = 1.5$
- This example shows the danger of using only one factor (instruction count) to assess performance - all three components must be considered when comparing machines on the basis of execution time

COMP3211/9211

2003 S2 L2 P53

## Improving performance

- Given an instruction set, CPU performance can be improved by
  - Increases in the clock rate
  - Improvements in processor organization that lower the CPI
  - Compiler enhancements that lower the instruction count or generate instructions with a lower CPI
- Improvements in one aspect may affect other aspects

COMP3211/9211

[Gur]

2003 S2 L2 P54

## Speedup

- Speedup due to enhancement  $E$ :

$$\text{speedup}(E) = \frac{\text{ex\_time}(\text{ })}{\text{ex\_time}(E)} = \frac{\text{performance}(E)}{\text{performance}(\text{ })}$$

COMP3211/9211

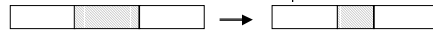
2003 S2 L2 P55

## Amdahl's Law

- Suppose that enhancement  $E$  accelerates a fraction  $F$  of the task by a factor  $S$  and the remainder of the task is unaffected. Then

$$\text{ex\_time}(E) = ((1-F) + F/S) \times \text{ex\_time}(\text{ })$$

$$\text{ex time after improv} = \frac{\text{ex time affected by improv}}{\text{amount of improv}} + \text{ex time unaffected}$$



- Speedup is limited by the amount that the improved feature is used

$$\text{speedup}(E) = \frac{1}{(1-F) + F/S}$$

COMP3211/9211

[Patterson]

2003 S2 L2 P56

## Example: Amdahl's Law

- Suppose a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds of this time. How much does the speed of the multiplication operation have to be improved for the program to run five times faster?

$$\frac{100}{5} = \frac{80}{x} + 20$$

$$x \rightarrow \infty$$

COMP3211/9211

[Guo]

2003 S2 L2 P57

## Next lecture

- Background Part 3 – VHDL

COMP3211/9211

2003 S2 L2 P58