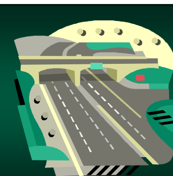
Design Principles

Part 9

1



von Neumann Architecture

The Information Superhighway

2

von Neumann Machine Architecture

- Modern computers are based on the design of John von Neumann
- His design greatly simplified the construction of (and use) computers

3

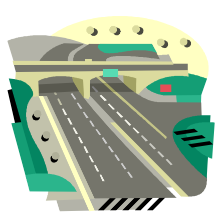
Some von Neumann Attributes

1. Programs are stored and executed in memory
2. Separation of processing from memory
3. Different system components communicate over a shared bus

4

The Bus

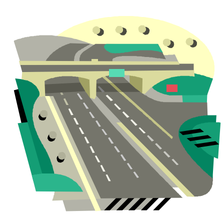
- Electronic pathway that transports data between components
- Think of it as a "highway"
  - data moves on shared paths
  - otherwise, the computer would be very complex



5

System Bus

- The information sent on the memory bus falls into 3 categories
- Three sets of signals
  - address bus
  - data bus
  - control bus



6

Address Bus

- Used by the processor to access a specific piece of data
- This "address" can be
  - a specific byte in memory
  - unique IO port
  - etc…
- The more bits it has, the more memory can be accessed

7

Address Bus Size Examples

- 8-bit → 2^8 = 256 bytes
- 16-bit → 2^{16} = 64 KB (65,536 bytes)
- 32-bit → 2^{32} = 4 GB (4,294,967,296 bytes)
- 64-bit → 2^{64} = 18 EB (18,446,744,073,709,551,616)

8

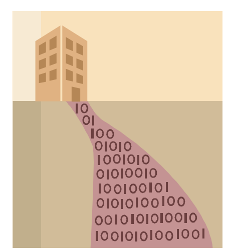
Historic Address Sizes

- Intel 8086
  - original 1982 IBM PC
  - 20-bit address bus (1 MB)
  - only 640 KB usable for programs
- MOS 6502 computers
  - Commodore 64, Apple II, Nintendo, etc…
  - 16-bit address bus (64 KB)

9

Data Bus

- The actual data travels over the *data bus*
- The number of bits that the processor uses – as its natural unit of data – is called a *word*



10

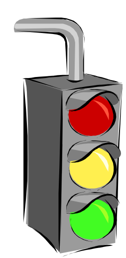
Data Bus

- Typically we define a system by word size
- Example:
  - 8-bit system uses 8 bit words
  - 16-bit system uses 16 bits (2 bytes) words
  - 32-bit system uses 32 bits (4 bytes) words
  - etc…

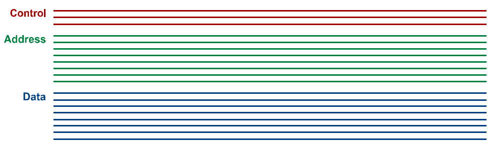
11

Control Bus

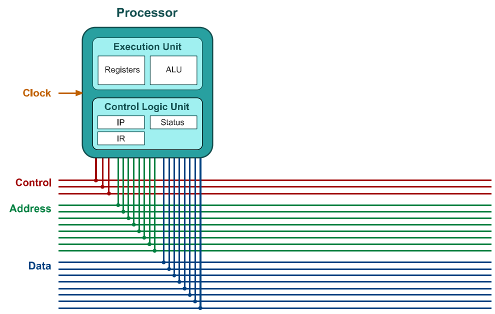
- The *control bus* controls the timing and synchronizes the subsystems
- Specifies what is happening
  - read data
  - write data
  - reset
  - etc…



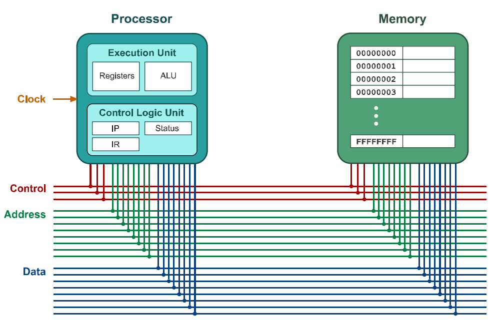
12



13



14



15

von Neumann Architecture Today

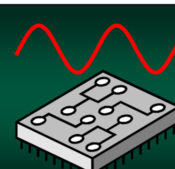
- Because of the emphasis on memory, most real-world systems use a modified version of his design
- In particular, they have a special high-speed bus between the processor and memory

16

von Neumann Architecture Today

- Think of it as a diamond-lane on a freeway
- … or as high-speed rail – which has a fixed source and destination and goes faster than the freeway

17



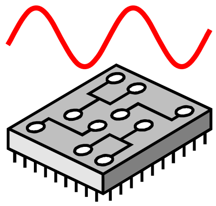
The System Clock

Tick-tock tick-tock tick-tock

18

The System Clock

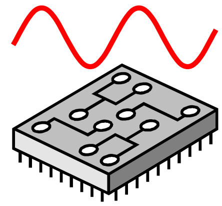
- The rate in which instructions are executed is controlled by the CPU clock
- The faster the clock rate, the faster instructions will be executed
- Measured in Hertz – number of oscillations per second



19

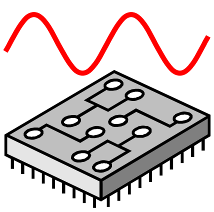
The Clock

- Computers are typically (and generically) labeled on the processor clock rate
- In the early 80's it was about 1 Megahertz – million clocks per second
- Now, it is terms of Gigahertz – billion clocks per second



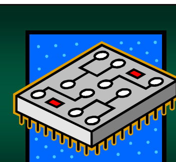
20

Clock and Instructions



- Not all instructions are "equal"
- Some require multiple clock cycles to execute
- For example:
  - add can take a single clock
  - but floating-point math could require a dozen

21



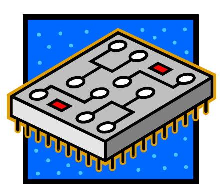
Technological Trends

Change is fast

22

Technological Trends

- Since the design of the integrated circuit, computers have advanced *dramatically*
- Home computer's today have more power than mainframes did 30 years ago
- A hand calculator has more power than the computer that took us to the Moon



23

Integrated Circuits Improved In…

- *Density* – total number transistors and wires can be placed in a fixed area on a silicon chip
- *Speed* – how quickly basic logic gates and memory devices operate
- *Area* – the physical size of the largest integrated circuit that can be fabricated

24

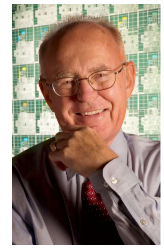
Rate of Improvement

- The increase in performance does <u>not</u> increase at a linear rate
- Speed & Density improves *exponentially*
  - from one year to the next… it has been a relatively constant fraction of the previous year's performance
  - …rather than constant absolute value

25

Moore's Law

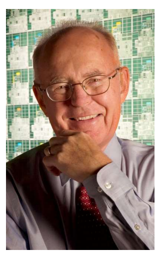
- Gordon Moore is one of the co-founders of Intel
- He first observed (and predicted) computer performance improves *exponentially*, not linearly



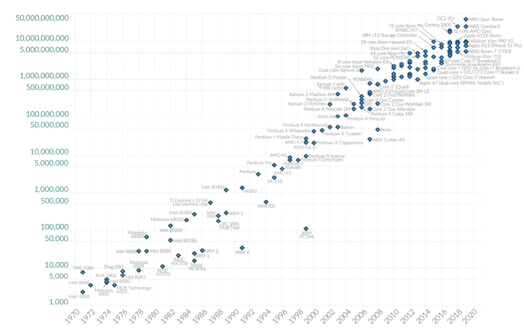
26

Moore's Law

- *Moore's Law* states the performance doubles every 18 months
- This law has held for nearly 50 years



27



28

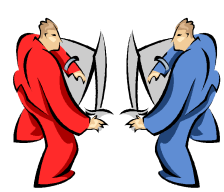
CISC vs. RISC

How do we tip the scales?

29

CISC vs. RISC

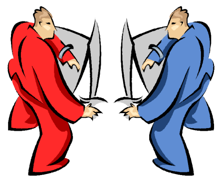
- There is, an often contentious, debate on how to design a processor
- For instance:
  - how is memory going to be accessed
  - what instructions are needed
  - how to encode/structure them



30

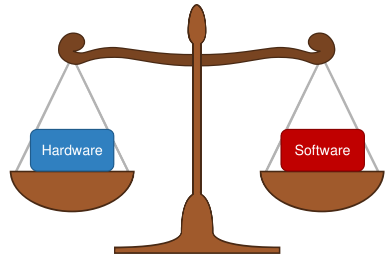
CISC vs. RISC

- Typically, the debate comes down to CISC vs. RISC
- Processors are typically put into these two categories
- Rarely is a processor "pure" RISC or CISC
- It's a *design philosophy* with a large "gray" area



31

Hardware vs. Software



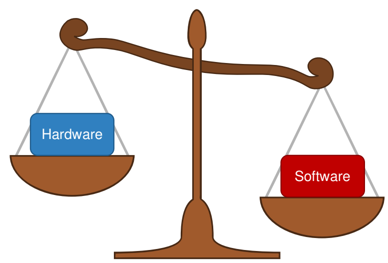
32

RISC

- Reduced Instruction Set Computer (RISC) emphasizes hardware simplicity
- Software should contain the complexity rather than hardware

33

RISC – Simple Hardware



34

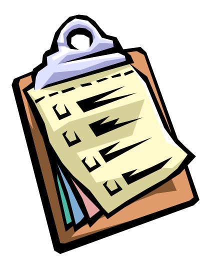
RISC

- So, RISC contains fewer instructions than CISC – only the minimum needed to work
- Minimalize memory access
  - only a few instructions can access memory
  - usually limited to register load and store instructions

35

RISC Characteristics

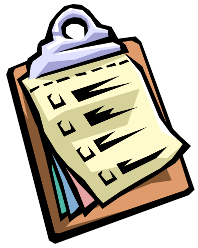
- Access to memory is restricted to load/store instructions
- Many registers – since all instructions can only use registers for calculations



36

RISC Characteristics

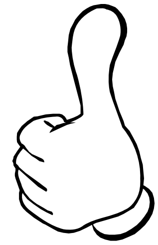
- Instructions typically take one clock cycle each
- The number of bytes, used by instructions, tend to fixed in size (all 32-bit, for example)



37

RISC Advantages

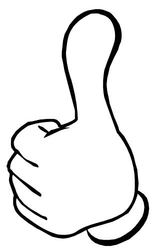
- Simpler instructions simplify hardware - makes processors easier to manufacture
- Also, produces less heat and requires less energy



38

RISC Advantages

- Fewer instructions means there is less to learn and master
- Memory access is minimalized



39

Example RISC Processors

- ARM
  - dominant processor used by smartphones
  - designed to reduce transistors
  - less cost, less heat, less power
- IBM PowerPC 601
  - developed in by IBM, Apple, and Motorola (AIM)
  - used by 1990's Macs

ARM POWERED

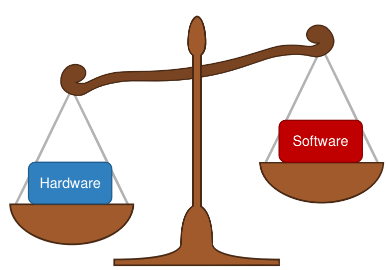
40

CISC

- Complex Instruction Set Computer (CISC) emphasizes flexibility in instructions
- Hardware should contain the complexity rather than the software

41

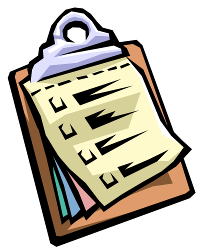
CISC – Hardware is more complex



42

CISC Characteristics

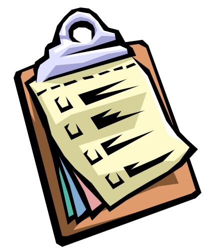
- Instructions can take multiple clocks – depending on how complex
- Operands are *generalized* – each can access memory, immediates or registers



43

CISC Characteristics

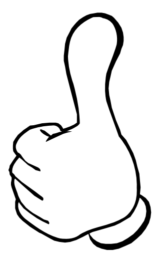
- Very few general-purpose registers
- The number of bytes, used by instructions, tend to vary in sizes



44

CISC Advantages

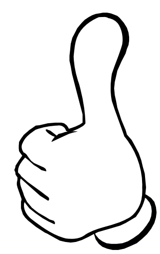
- Generally, requires fewer instructions than RISC to perform the same computation
- Software is easier to write given the flexibility



45

CISC Advantages

- Programs written for CISC architectures tend to take less space in memory
- Variable-sized instructions can make it possible for the processor "evolve" – i.e. add new instructions



46

Example CISC Processors

- Intel x86
  - evolved from the 8088 processor and contains 8-bit, 16-bit, and 32-bit instructions
  - dominant processor for PCs
- Motorola 68000
  - used in many 80's computers
  - …including the first Macintosh

47

Example CISC Processors

- VAX
  - contained even more addressing modes than we will cover
  - specialized instructions – even case blocks!
  - supported data types beyond float and int: variable-length strings, variable-length bit fields, etc…

48

RISC vs. CISC Comparison

CISC	RISC
Simple software, complex hardware	Simple hardware, complex software
Most operands can access memory	Load/Store instructions can access memory
Low number of registers	Higher number of registers
Instructions can have multiple clock cycles	Instructions tend towards one per clock cycle
Encoded instructions vary in size	Encoded instructions are all the same size

49

CISC Example (not x86)

```
# n  =  a * (b + c) - d

mov    R1, b
add    R1, c        # b + c

mul    R1, a        # a * (b + c)
sub    R1, d        # a * (b + c) - d

mov    n, R1
```

Note that ADD both loads & adds. This is 2 operations.

These too!

50

RISC Example (not x86)

```
# n  =  a * (b + c) - d

load   R1, b
load   R2, c
add    R2, R1      # b + c
load   R3, a
mul    R2, R3      # a * (b + c)
load   R4, d
sub    R2, R4      # a * (b + c) - d
store  n, R2
```

Note that all instructions (besides load & store) have two registers as operands.

51

Moore's Law & CISC

The pendulum swings

52

Moore's Law & CISC

- In the early 80's (the beginning of the Computer Revolution), memory was *expensive*
- In fact, it was the most expensive part of a new computer

53

Moore's Law & CISC

- Memory also ran at the same speed as the processor
- So, CISC was at a clear advantage – programs didn't take up as much memory as RISC

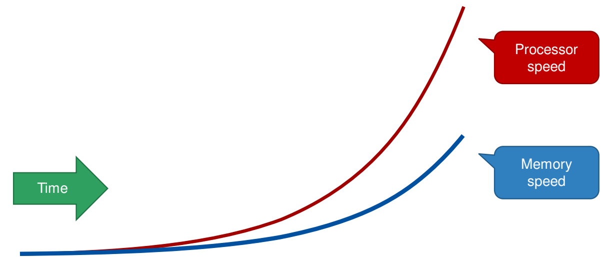
54

Moore's Law & CISC

- Computer speed through the 1980's grew exponentially
- However, …
  - rate of processor growth has been far greater than memory
  - so, memory *relative to the processor's speed* has gotten much slower

55

Moore's Law During the 80's



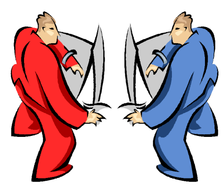
56

Memory is the Bottleneck

- CISC can access memory with nearly every instruction
- Memory is slow compared to register-to-register operations
- It is far more efficient (now) to do all work on the processor and use memory **only** when absolutely necessary

57

Latest Approach



- After the 1990s, RISC architectures have incorporated some of most useful complex instructions from CISC architectures
- Rely on micro-architecture to implement these instructions with little impact on the clock

58

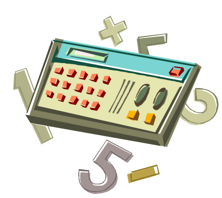
Instruction Operands

How much data does each need?

59

Instruction Operand

- The number of operands used in an instruction varies greatly by processor
- More operands give greater functionality, but require more bits to store in memory
- Typically processors contain 1, 2 or 3 operands



60

Single Operand Processors

- Single operand processors are also known as *accumulators*
- Operates similar to your hand calculator
- The accumulator register
  - used for all mathematical computations
  - other registers simply are used to compare and hold temporary data
- Examples: MOS 6502

61

Single Operand Instruction (CISC)

```
#  z  =  50 - (x + y)

lda  x
add  y          # x + y
sta  temp
lda  50
sub  temp       # 50 - temp

sta  z
```

lda = Load accumulator

sta = store accumulator

62

Two Operand Processors

- Allows two operands to be specified
- For computations, both operands are typically treated as input, and one is used to store the result
- Examples:
  - x86 processors
  - PowerPC

63

Two Operand Instruction (CISC)

```
#  z  =  50 - (x + y)

mov   R1, x
add   R1, y    # x + y

mov   R2, 50
sub   R2, R1   # 50 - R1

mov   z, R2
```


64

Three Operand Processors

- Allows two input values like before, but also can specify a third output operand
- The third operand can also be used as a index for simple addressing
- Examples:
  - ARM
  - Intel Itanium

65

Three Operand Instruction (RISC)

```
#  z  =  50 - (x + y)

load  R1, x
load  R2, y
add   R3, R1, R2   # x + y
load  R4, 50
sub   R5, R4, R3
store z, R5
```


66

Three Operand Instruction (CISC)

```
#   z  =  50 – (x + y)

add  R1, x, y         # x + y
sub  z, 50, R1
```

Best of both worlds!

67