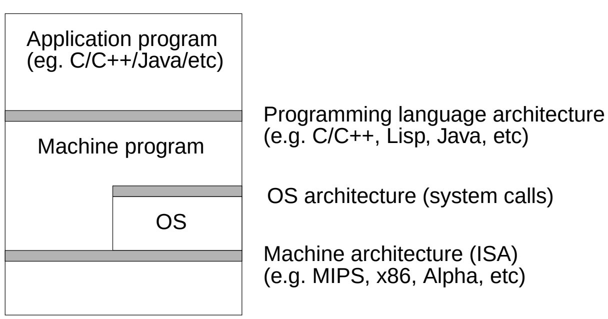
Course Introduction

What this course is about

- *Hardware/Software interface:*
  - Compilers, assemblers, linkers, loaders: who does what in terms of getting my program to run?
  - What kind of instructions does the machine understand?
- *Organization*:
  - What are the basic pieces of the machine (registers, cache, ALU, busses)?
  - How are these pieces connected? How are they controlled?
- *Performance*:
  - What does it mean for one machine to be “faster” than another?
  - What are MFLOPS, MIPS, benchmark programs?

Levels of Abstraction

- We can describe a computer system as a set of layers:



Instruction Set Architecture

- ISA is an interface between the hardware and software.
- ISA is what is visible to the programmer (note that the OS and users might have different view)
- ISA consists of
  - instructions (operations, how are they encoded?)
  - information units (what is their size, how are they addressed)
  - registers (general or special purpose)
  - input-output control
- ISA is an abstract view of the machine: underlying details should be hidden from the programmer (although this is not always the case)

Computer Families

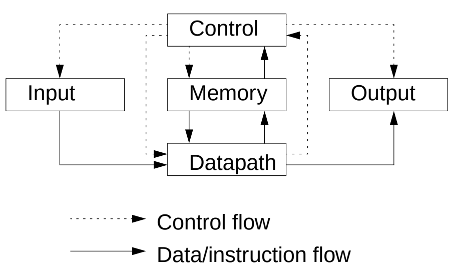
- Sequence of machines that have the same ISA (binary compatible). For example:
  1. IBM 360 Series (invented the notion of ISA in 1960s)
  2. DEC PDP-11, VAX [1970s]
  1. Intel x86 (80386, 80486, Pentium, PII, PIII, PIV)
  2. Motorola 680x0
  3. MIPS Rx000 [1980s to present]
  4. Sun SPARC [1980s to present]
  5. DEC Alpha (21x64) [1990s to present]
- With “portable” software, are “binary compatible” machines important?

Computer Generations

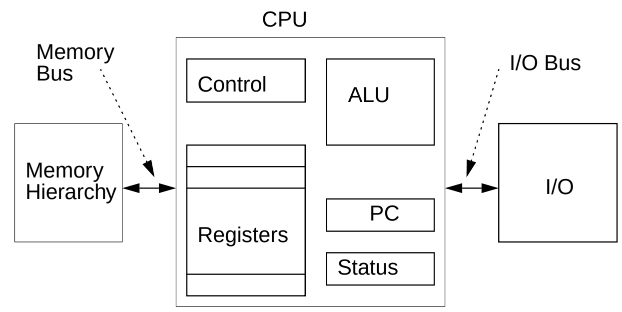
	1st	2nd	3rd	4th	5th	...
Processor Technology	Vacuum tubes	transistors	integrated circuits	LSI	VLSI	Very VLSI
Processor Structure	single processor	multiple functional units	micros and minis	workstations and PCs	32-bit microcomputers	64-bit + MP micros
Memory	Vacuum tubes	Magnetic core	semiconductors	semicond. 64KB	semicond. 512 KB	semicond. 64 MB
Example machine	UNIVAC 1950s	Burroughs 5500 1960-68	PDP-11 1969-77	Apple II 1978-mid 80s	Apple Mac, 1980s	Alpha, SPARC, 1990s

Stored Program Computer

- Instructions and data are binary strings
- 5 basic building blocks: arithmetic (datapath), control, memory, input, output:



Computer Structure



The CPU - What does it do?

- The CPU “executes” the following program:

```
while (TRUE) do
    fetch the next instruction
    decode it
    execute it
    calcluate the address of the next instruction
end while
```

- How does it know where to find the next instruction?
- Where does it “keep” the current instruction?
- Where do instructions come from?
- When does it stop?
- We’ll be refining this picture during the next few weeks....

Instructions

- An instruction tells the CPU:
  - The operation to be performed (the opcode)
  - The operands (zero or more)
- For a given instruction, the ISA specifies
  - the meaning (semantics) of the opcode
  - how many operands are required (and their types)
- Operands can be of the following type
  - registers
  - memory address
  - constant (immediate data)
- In MIPS, the operands are typically registers or small constants

Registers

- Registers are visible both to hardware and programmer
  - High-speed storage of operands
  - Easy to name
  - Also used to address memory
- Most current computers have 32 or 64 registers
- Not all registers are “equal”
  - Some are special purpose (eg. in MIPS $0 is hardwired to 0).
  - Integer / Floating point
  - Conventions (stack pointers)
- Why no more than 32 or 64? (at least 3 good reasons)

The Memory System

- Memory is a hierarchy of devices/components which get increasingly faster (and more expensive) as they get nearer to the CPU:

Memory level	**Capacity (bytes)**	**Speed**	**Relative Speed**	**Price**
Registers	1000s	nanoseconds	1	??
Cache	16KB on-chip 1MB off-chip	nanoseconds 10s of ns	1-2 5-10	?? $100/MB
Primary memory	10-100MB	10s to 100s ns	10-100	$1/MB
Secondary mem.	1-10GB	10s of ms	1,000,000	$.01/MB

- Library metaphor of memory hierarchy

Memory

- Memory is an array of information units
  - Each unit has the same size
  - Each unit has a unique address
  - Address and contents are different

A memory of size N units

Address	Contents
Address 0	122
1	-4
2	14
n-1	

- A C variable is an abstraction for a memory location

Information Units

- Basic unit is the *bit* (stores a 0 or a 1)
- Bits are grouped together into larger units:
  - bytes = 8 bits
  - words = 4 bytes
  - double words = 2 words (8 bytes)

Binary Representation

- Computers represent all data (integers, floating point numbers, characters, instructions, etc.) in a binary representation. *Interpretation depends on context.*
- Know your (common) powers of two!

Power	Value	Slang
8	256	...
10	1024 or ~1000	1K
16	65536 or ~64000	64K
20	~1,000,000	1M
30	~1,000,000,000	1G
32	~4,000,000,000	4G

2s Complement

- Representing integers: What characteristics does our scheme need?
  - Easy test for positive/negative.
  - Equal number of positive and negative numbers
  - Easy check for overflow
- Different schemes: sign and magnitude, 1's complement, 2's complement
- 2's complement tricks (sign bit extension, converting from positive to negative, addition/subtraction)
- Modern machines use 2s complement
- 2s complement numbers are easy to add and negate, giving us subtraction for "free"
- 2s complement tricks: sign extension, negation, addition/ subtraction
- Hexidecimal notation

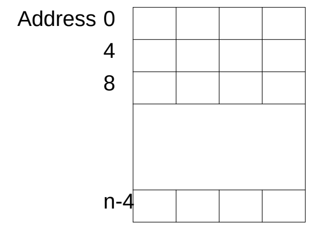
Addressing

- The *address space* is the set of all information units that a program can reference
- Most machines today are *byte addressable*
- Processor "size" impacts the size of the address space:
  - 16 bit processor: 64KB (too small nowadays)
  - 32 bit processor: 4GB (starting to be too small)
  - 64 bit processor: really big (should last for a while...)
- Rule of thumb: We're using up address space at a rate of around 1 bit per year...

Addressing Words

- On a byte addressable machine, every word starts at an address divisable by 4:

A memory of size N bytes



- Big vs. Little Endian: within a data unit (eg. word), how are the individual bytes laid out?
- Little/Big: address of data unit is address of low/high order byte (DEC MIPS is Little; SGI MIPS, SPARC are Big)