

Computer Basics



Regardless of how much computers have changed over the last 50 years (think of our first lecture), they are still characterized by the same basic principles

1/30/2002

CB-1

© Copyright 2000-2002, University of Washington



Abstractly, A Computer Is...

- ∇ Computers process information by deterministically following instructions, called *executing* instructions
- ∇ Unlike humans, computers follows instructions *exactly*
 - Computers have no imagination or creativity
 - Computers have no intuition
 - Computers are literal: they have no sense of irony, subtlety, proportion...
 - Computers don't joke, they're not vindictive or cruel
 - Computers are not purposeful (they don't have their own changing agenda!)

...Computers execute instructions. Nothing more.

1/30/2002

CB-2

© Copyright 2000-2002, University of Washington



Remember this when you feel like screaming at your monitor....!

If a computer has any useful characteristics, it's because someone has programmed it –in other words, given it the instructions – to behave usefully

1/30/2002

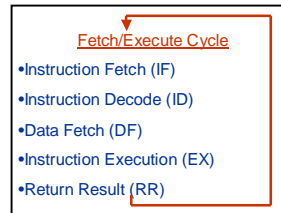
CB-3

© Copyright 2000-2002, University of Washington



Interpreting the Instructions

- ∇ To perform instructions, a computer's hardware implement a process called the *fetch/execute cycle*



The F/E Cycle is an unending process

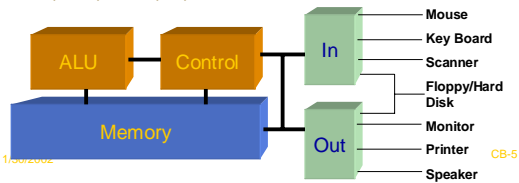
1/30/2002

CB-4

© Copyright 2000-2002, University of Washington

Anatomy Of A Computer

- ∇ A computer is essentially made up of 5 components:
 - Arithmetic/Logic Unit (ALU) – the part doing computations
 - Control – the part that follows the Fetch/Execute Cycle of the program and tells the ALU what to compute
 - Memory – where data, programs are kept while computing
 - Input – ports to peripheral devices that allow/bring data in
 - Output -- ports to peripheral devices that allow/send data out



1/30/2002

CB-5

A simple example

- ∇ Suppose you have
 - A set of envelopes, each with a card in it
 - A number or an instruction can be written on each card
 - There are three kinds of instructions:
 - ADD env# env# env#
 - ASK env#
 - SAY env#
 - NEXT env#

1/30/2002

CB-6

A simple example

Envelope 1: ASK 15
 Envelope 2: ASK 13
 Envelope 3: ADD 15 13 10
 Envelope 4: SAY 10
 Envelope 5: NEXT 1
 Envelope 10: ??
 Envelope 13: ??
 Envelope 15: ??

1/30/2002

CB-7

A simple example

Envelope 1: ASK 15
 Envelope 2: ASK 13
 Envelope 3: ADD 15 13 1
 Envelope 4: SAY 10
 Envelope 5: NEXT 1
 Envelope 10: ??
 Envelope 13: ??
 Envelope 15: ??

1/30/2002

CB-8

FIT 100 Memory

- ▼ The memory component is passive, storing programs and data

address: 0 1 2 3 4 5 6 7
 value:

M	J	i	s	!	23	2	3
---	---	---	---	---	----	---	---

↔
byte

- ▼ Memory is like a series of “byte-size” boxes – each has an **address** and some contents called its value
- ▼ Memory is called **RAM** for “random access memory” because the control can access any random location in the memory
- ▼ RAM is volatile memory – it disappears when the power does

1/30/2002 CB-9
 © Copyright 2000-2002, University of Washington

FIT 100 There always needs to be something in Control: Control Rules!

- ▼ The control follows through the instructions, executing them by telling other parts what to do
- ▼ The instructions come from the program stored in the memory

The instructions are in the end expressed in a *machine language*, which the control can understand. A typical machine instruction is

add 124, 1005, 6215

Which means “add the number in memory location 124 To the number in memory location 1005 and put the result in memory location 6215”

1/30/2002 CB-10
 © Copyright 2000-2002, University of Washington

FIT 100 Just to be clear...

- ▼ The instruction **add 124, 1005, 6215** does not add 124, 1005 and 6215 together. We can do that in our heads or with a calculator
- ▼ It simply adds whatever has been stored at those memory locations
- ▼ Different numbers in those locations produce different results:

add 124, 1005, 6215

124	+	1005	→	6215
23		2	→	25
124	+	1005	→	6215
0		35	→	35
124	+	1005	→	6215
699		-2	→	697

1/30/2002 CB-11
 © Copyright 2000-2002, University of Washington

FIT 100 Following Instructions

- ▼ The control maintains the correct place in the program by using a **program counter**, or PC. A better name might be “instruction pointer”.
- ▼ The control also prepares for data-fetches from and result-returns to the memory

PC: program counter, personal computer and printed circuit board

Memory

...	add 124, 1005, 6215	...
-----	---------------------	-----

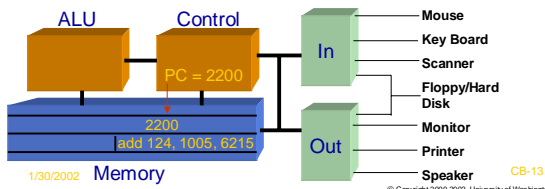
- Fetch instruction from memory at PC
- Decode the Instruction; PC ← PC + 1
- Get Data needed for Instruction
- Execute (perform) instruction
- Return Result to Memory

1/30/2002 12
ington

FIT 100

The Fetch/Execute Process

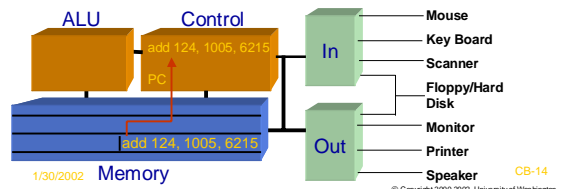
- Just before the Instruction Fetch....



FIT 100

Instruction Fetch

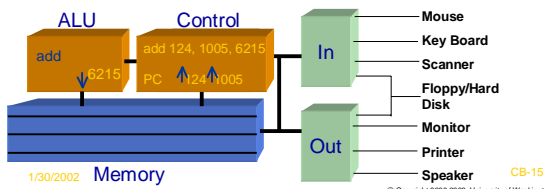
- Get instruction at the memory location PC



FIT 100

Instruction Decode

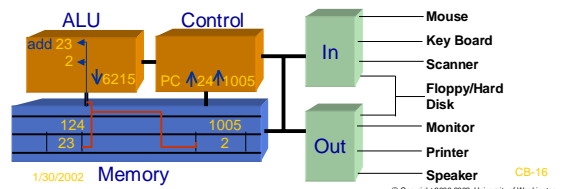
- Analyze Instruction and set up later steps
 - Specify the ALU operation (add)
 - Specify addresses to fetch (124, 1005) and to store (6215)



FIT 100

Data Fetch

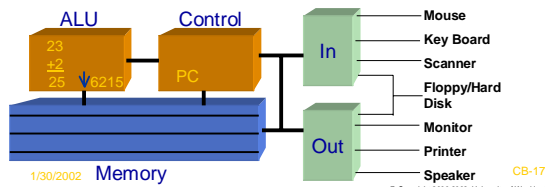
- Move values stored at fetch-addresses to ALU for processing



FIT 100

Execute

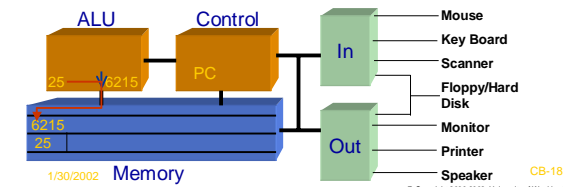
- ▼ The operation of the instruction (add) is performed



FIT 100

Result Return

- ▼ The result is returned to memory to the address specified in the instruction



FIT 100

The PC's PC

- ▼ After the instruction has been fetched and executed, the next instruction in sequence is fetched at PC +1
- ▼ This scheme should cause the computer to run through memory executing all instructions once and then "fall off the end of memory"
- ▼ Computers have machine instructions to **branch** and **jump**, i.e. go to some instruction other than the next
- ▼ **Jump** and **Branch** change the PC after increment
- ▼ Programs generally repeat many instructions

CB-19

© Copyright 2000-2002, University of Washington

FIT 100

What's in a Number?

- ▼ A memory location can store one byte of information, enough for a keyboard character
- ▼ A "normal" whole number (integer) uses 4 bytes
- ▼ A machine instruction uses 4 bytes
- ▼ Units of memory size are ...
 - KB, **kilobyte**, 1,024 bytes ... just over a thousand bytes, a "K"
 - MB, **megabyte**, 1,048,576 bytes ... just over a million bytes, a meg
 - GB, **gigabyte**, 1,073,741,824 bytes ... just over a billion bytes, a "gig"
 - TB, **terabyte**, 1,099,511,627,776 bytes ... just over a trillion bytes

1/30/2002

CB-20

© Copyright 2000-2002, University of Washington

**FIT
100**

Free Memory!

- ▽ Why do computers use such weird amounts to indicate 1000, 1 Million, etc?
 - These numbers are powers of 2
 - $2^{10} = 1,024$ call it a thousand
 - $2^{20} = 1,048,576$ call it a million
 - $2^{30} = 1,073,741,824$ call it a billion
 - $2^{40} = 1,099,511,627,776$ call it a trillion
- ▽ When you buy a megabyte of member, it's as if you get 48, 576 bytes for free!

1/30/2002

CB-21

© Copyright 2000-2002, University of Washington

**FIT
100**

Computational Time: The Pace of Computing

- ▽ Computers use electronic clocks to pace the Fetch/Execute Cycle
- ▽ If the computer goes around the F/E cycle once per tick, then the rate of the clock ("ticks/second") gives the number of instructions executed per second
- ▽ Hertz measures "cycles per second"
- ▽ 500MHz, specifies "500 million cycles per second"
- ▽ The reality is that the "one instructions per clock cycle" rule is only an approximation... modern computers are MUCH more complicated

1/30/2002

CB-22

© Copyright 2000-2002, University of Washington

**FIT
100**

Summary

- ▽ Computers deterministically execute instructions to process information
- ▽ Computers have five parts: ALU, Control, Memory, Input and Output
- ▽ The control implements a process called the Fetch/Execute Cycle
- ▽ The F/E cycles is a fundamental method of performing operations EXACTLY the same way specified, every time. This idea is used in many places in computation

1/30/2002

CB-23

© Copyright 2000-2002, University of Washington

**FIT
100**

For Monday

- ▽ Assignment 2 is due in your Monday/Tuesday lab
- ▽ Read Chapter 10 of FIT Course pack
- ▽ Lab 7 is the Introduction to Visual Basic
 - Read through the Lab
 - Read the Chapters suggested there.

1/30/2002

CB-24

© Copyright 2000-2002, University of Washington