General information

Course web page:

http://www.cs.bgu.ac.il/~caspl152

Instructors:

Prof. Eyal Shimony (course coordinator)

Office hours: Wed. 12-14 (for now)

Building 37 (Alon High-Tech), Room 216

Mr. Tamir Grossinger

TAs: Marina Kogan, Pavel Vaks

Lab TAs:

Ehud Apsel, Ehud Barnea, Itai Azaria,
Jumana Nassour, Majid Kassis,
Benny Lutati, Lior Lotan, Rafi Cohen,
Yonatan Svirsky, Daniel Zatulovsky,
Mark Abashkin, Husein Othman, Abed Asi.

Syllabus: (see web page)

Goals and Expectations

Architecture and Assembly Language

- Computer organization:
  - Basic Principles
  - Case study: 80X86
- Computer architecture:
  - Principles
  - Case study: 80X86
- Assembly and machine language
  - Principles
  - HANDS ON experience: 80X86
  - Integration and applications

SP lab

- Low-level systems-related programming via hands-on experience
- **Really** understanding data

Learning how to RTFM

SP Lab Issues

- Programming in C: understanding code and data (including pointers).
- Binary files: data structures in files, object code, executable files (ELF).
- System calls: process handling, input and output. Direct system calls.
- Low-level issues in program developement: debugging, patching, hacking.

Done through:

- Reasoning/exploration from basic principles.
- Implementation of small programs (in C).
- Interacting with Linux OS / systems services.

IMPORTANT Lessons

At the end of the course, Only REALLY need to KNOW* two things:

1) How to RTFM

2) There is no magic**

* KNOW: in "intelligent agent behaviour consistent with knowledge" meaning.

** Ref: Pug the magician

Why Bother?

Why bother? All software today is in JAVA

or some other HLL anyway?

- Essential for understanding (lower level of) COMPILERS, LINKERS, OS.
- Architecture has impact on performance. Writing a program for better PERFORMANCE, even in a HLL, requires understanding computer architecture.
- Some EMBEDDED CPUs: only assembly language available
- Some code (part of the OS) STILL done in assembly language.
- Better understanding of security aspects.
- Viruses and anti-viruses.
- Reverse engineering, hacking, and patching.
- **Everything** is data.

Role of Course in Curriculum

- Understanding of PHYSICAL implementations of structures from data-structures course.
- Can be seen as high-level of ``Digital Systems'' course.
- Understanding of computer operation at the subsystem level.
- Leads up to ``Compilers'' and “Operating Systems” as an ``enabling technology''
- Compilers course - compilers use assembly language or machine code as end product.
- Systems programming – the programmer’s interface to the OS.

Course outline

LECTURES (including SPlab (*))

1) *Introduction to course and labs (week 1)
2) Basic architecture and LOW-LEVEL programming issues. (weeks 1-7)
3) *Linux system services, shell (Week 8)
4) Assembly programming (weeks 9-10)
5) *ELF format, linking/loading (week 11)
6) Advanced LOW-LEVEL prog. (wks 12-14)
7) Intro to communication (weeks 14-15)

LABS:

- Simple C programs (weeks 1-6)
- System calls (weeks 6-7)
- Command interpreter (weeks 8-11)
- Handling ELF files (weeks 12-14)

Programmer's View of Computing

To program a computer:

1. Write a program in a source language (e.g. C)
2. COMPILER converts program into MACHINE CODE or ASSEMBLY LANGUAGE
3. ASSEMBLER converts program into MACHINE CODE (object code file)
4. LINKER links OBJECT CODE modules into EXECUTABLE file
5. LOADER loads EXECUTABLE code into memory to be run

Advanced issues modify simplified model:

1. Dynamic linking/loading
2. Virtual memory

Program Execution Basics (von-Neumann Architecture)

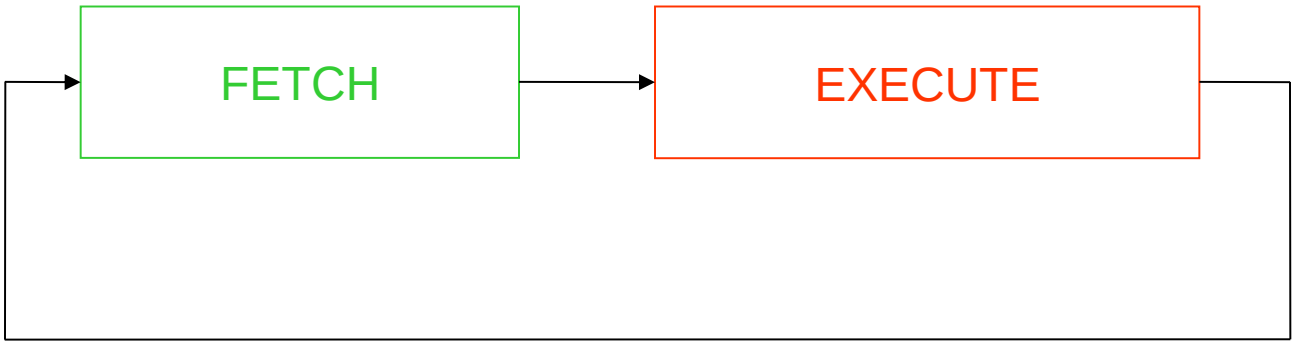
Computer executes a PROGRAM stored in MEMORY.

Basic scheme is - DO FOREVER:

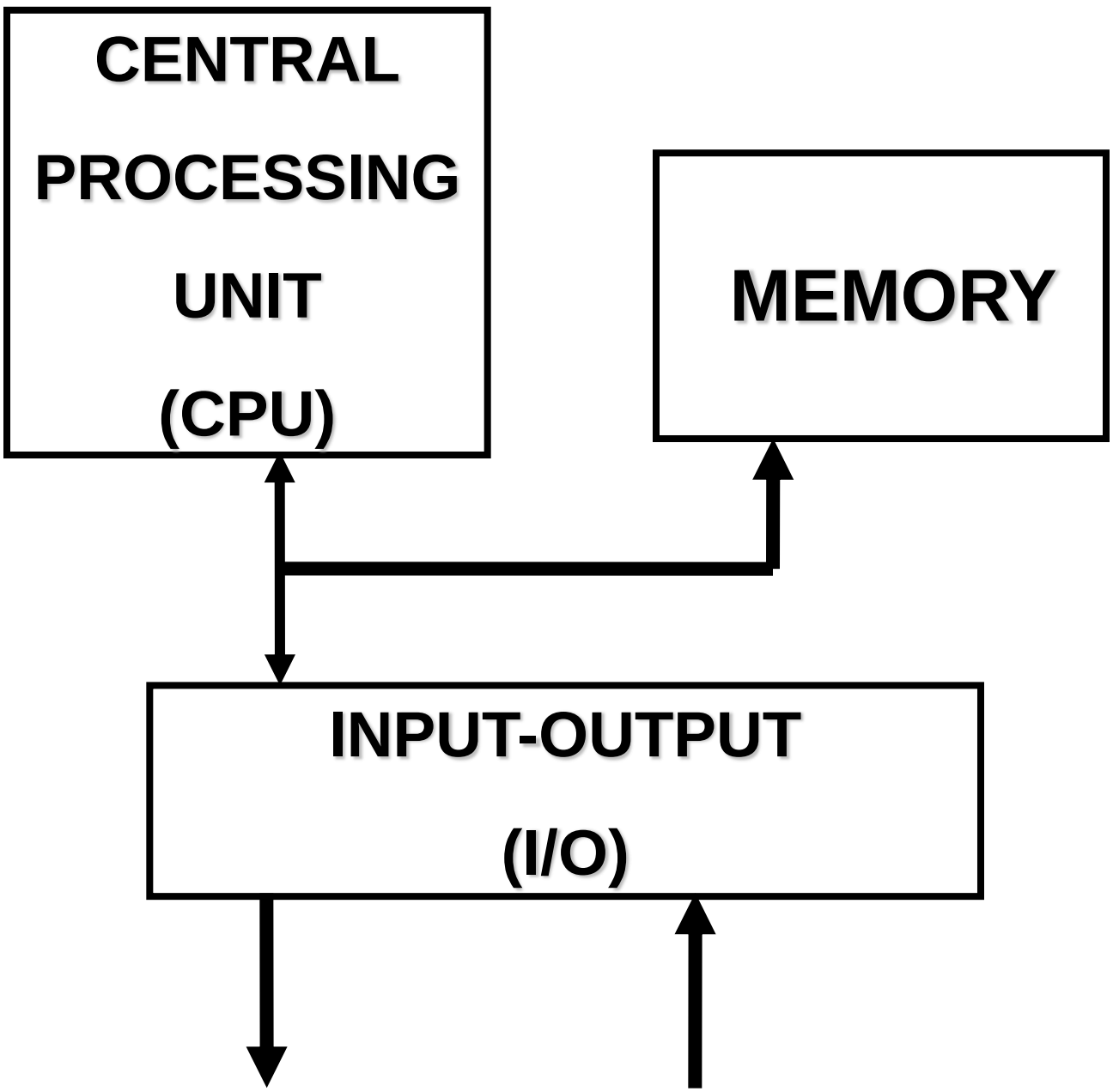
1. FETCH an instruction (from memory).
2. EXECUTE the instruction.

This is the FETCH-EXECUTE cycle.

More complicated in REAL machines (e.g. interrupts).



Block Diagram of a Computer

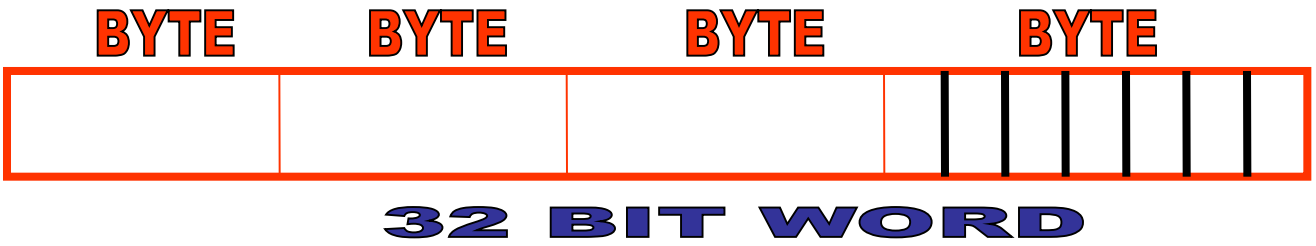


Data Representation Basics

Bit - the basic unit of information: (true/false) or (1/0)

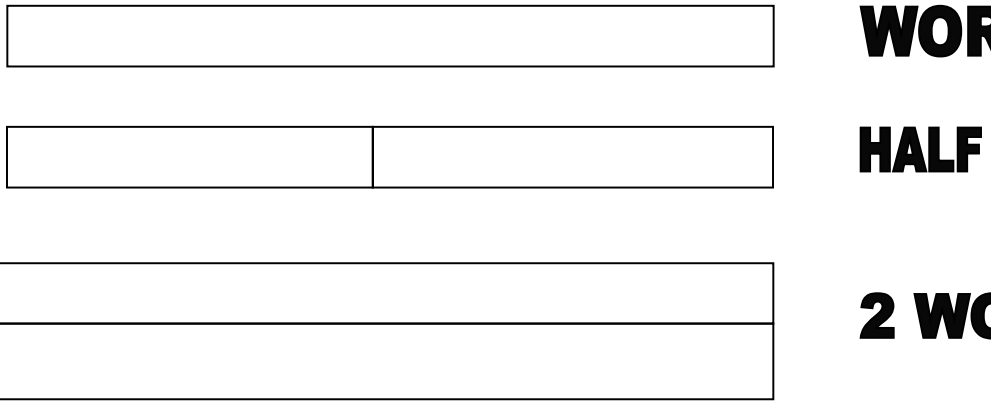
Byte - a sequence of (usually) 8 bits

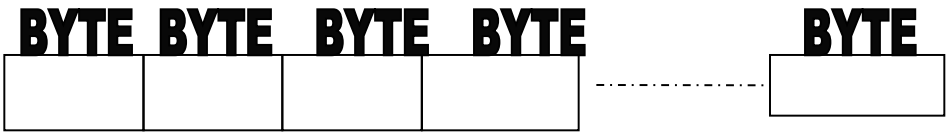
Word - a sequence of bits addressed as a SINGLE ENTITY by the computer (in various computers: 1, 4, 8, 9, 16, 32, 36, 60, or 64 bits per word)



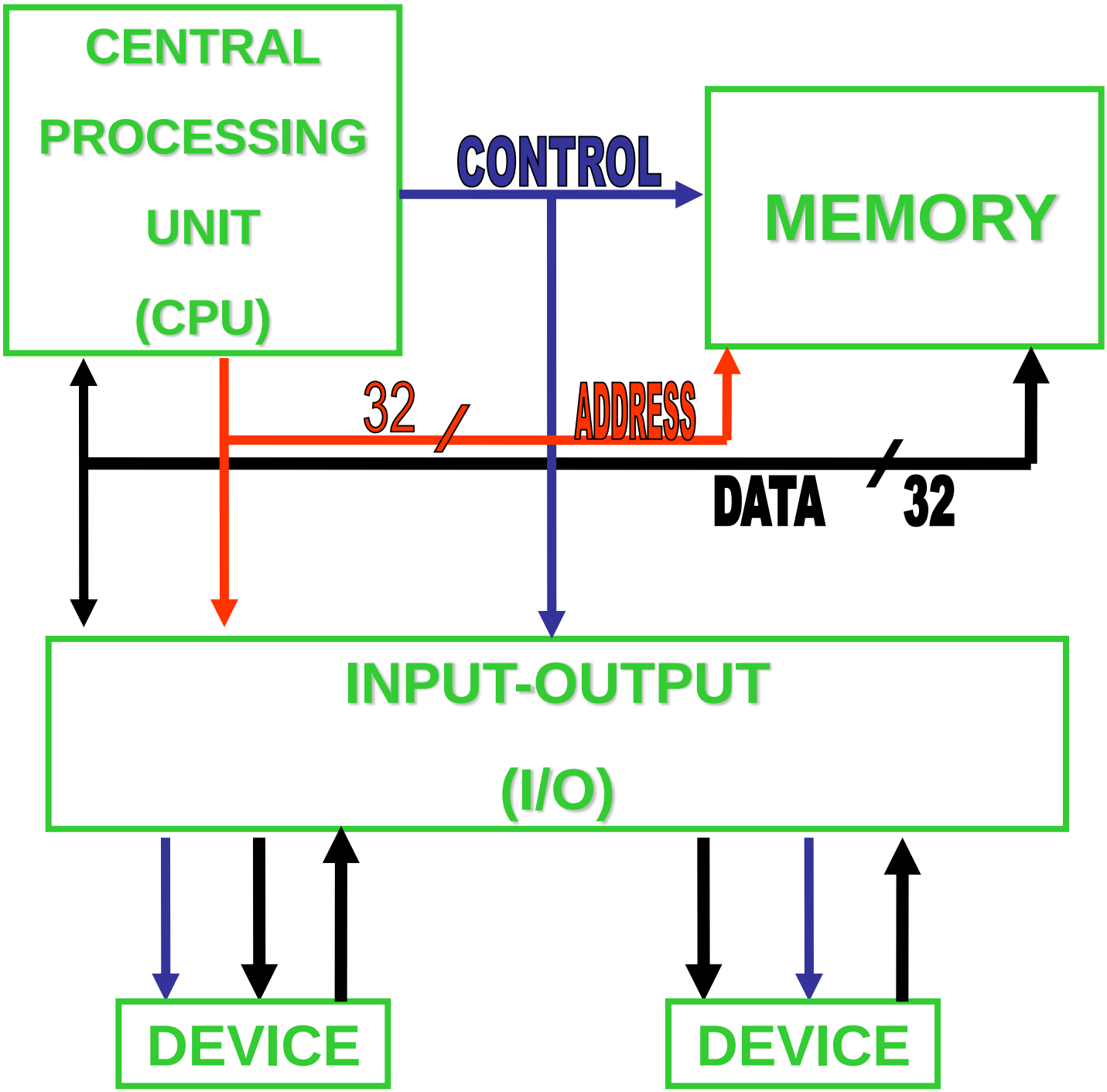
Character 6-8 bits (ASCII), 2 bytes, etc.

Instructions?





Refined Block Diagram



Basic Principles: Address Space

Physical (meaningful) addresses