General information

Course web page:

http://www.cs.bgu.ac.il/~caspl212

Instructors:

Prof. Eyal Shimony (course coordinator)

Office hours: Tuesday 14-16

(Zoom meeting number 260 939 8677)

Building 37 (Alon High-Tech), Room 216

Mr. Tamir Grossinger

Head TA:

Mr. Tamir Grossinger

Head Lab TA:

Mr. Tamir Grossinger

Course format:

Usually 2 hours lecture,

1 hour practical session,

3 hours SP lab.

Syllabus: (see course 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

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 development: debugging, patching, hacking.

Done through:

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

IMPORTANT Lessons

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

1) How to RTFM

2) There is no magic**

3) How to see things ***

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

** Ref: Pug the magician

*** Ref: Carlos Castaneda

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 1-6)
3) *Linux system services, shell (Week 7)
4) Assembly programming (weeks 8-9)
5) *ELF format, linking/loading (week 10)
6) Advanced LOW-LEVEL prog. (wks 11-13)
7) Input-output (week 14)

LABS:

- Simple C programs (weeks 2-4)
- System calls (weeks 5-6)
- Command interpreter (weeks 8-10)
- Handling ELF files (weeks 11-14)

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)

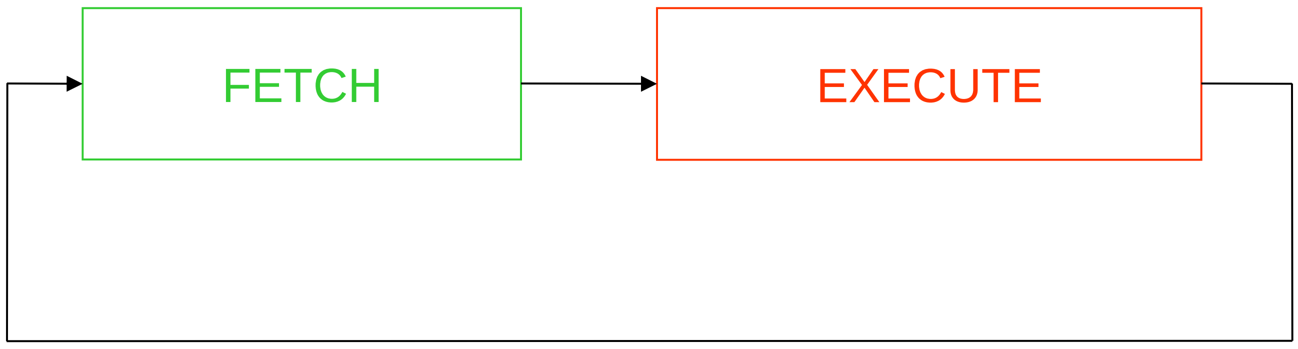
Computer executes a PROGRAM stored in MEMORY.

Basic scheme is - DO FOREVER:

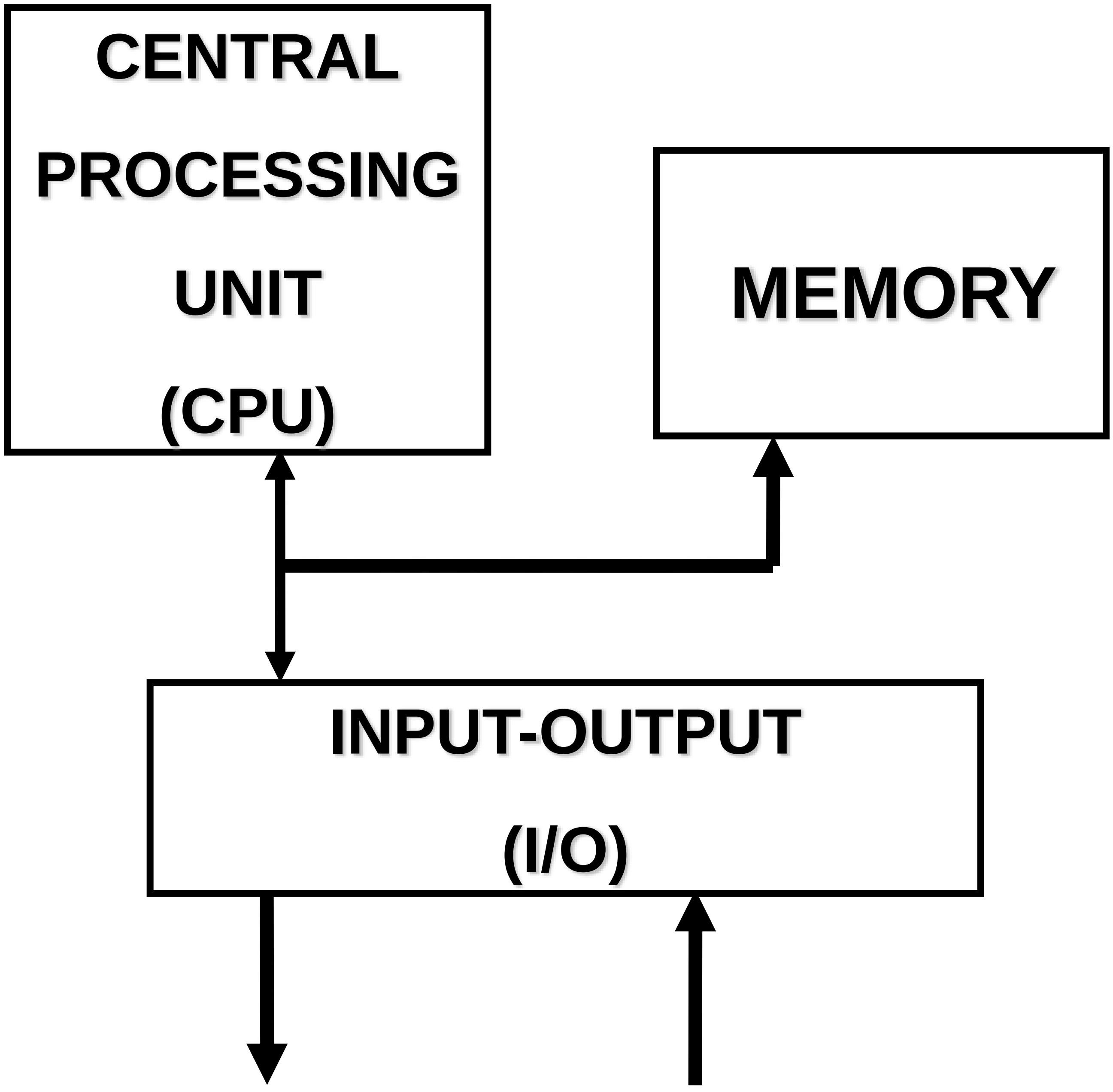
1. FETCH an instruction (from memory).
2. EXECUTE the instruction.

This is the FETCH-EXECUTE cycle.

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



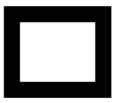
Block Diagram of a Computer



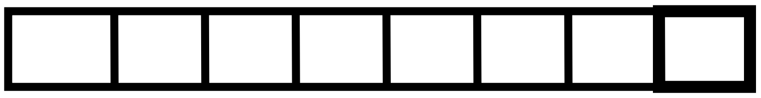
Data Representation Basics

Bit - the basic unit of information:

(true/false) or (1/0)

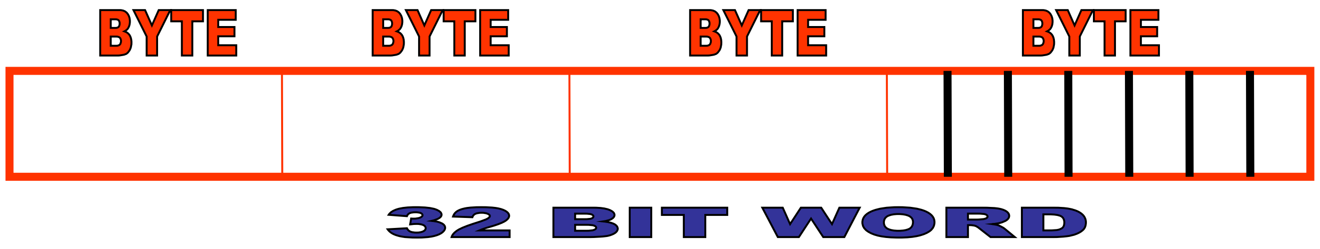


Byte - a sequence of (usually) 8 bits



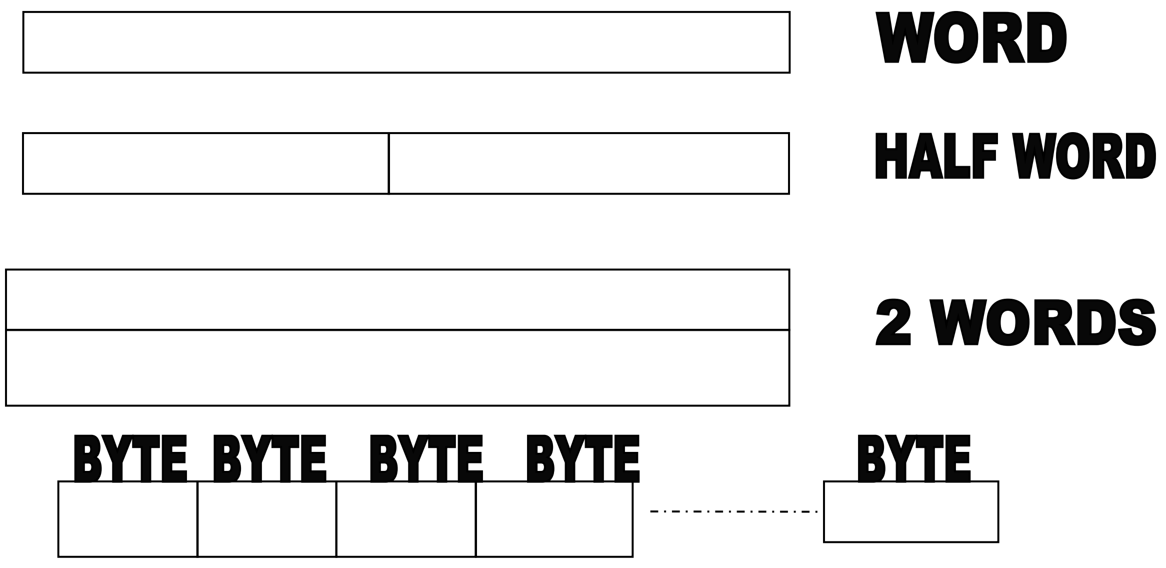
Word - a sequence of bits addressed as a SINGLE ENTITY by the computer

(in various computers: 1, 4, 8, 9, 16, 32, 36, 60, or 64 bits per word)

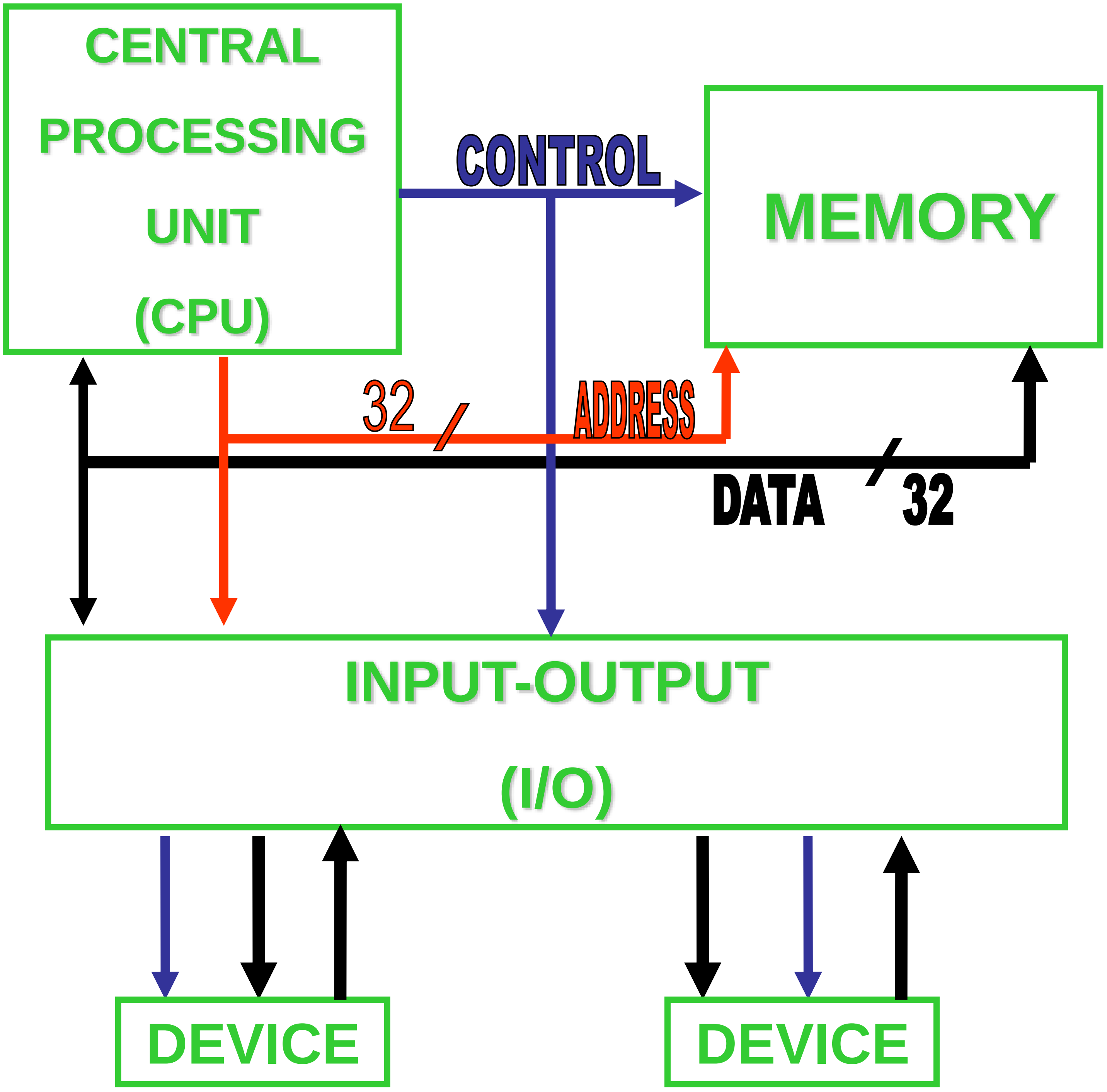


Character 6-8 bits (ASCII), 2 bytes, etc.

Instructions?

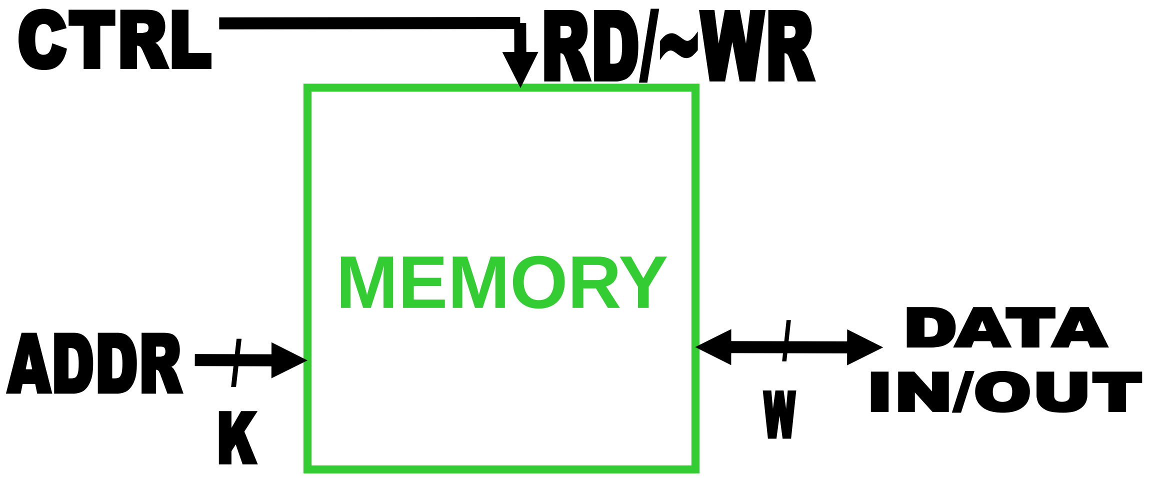


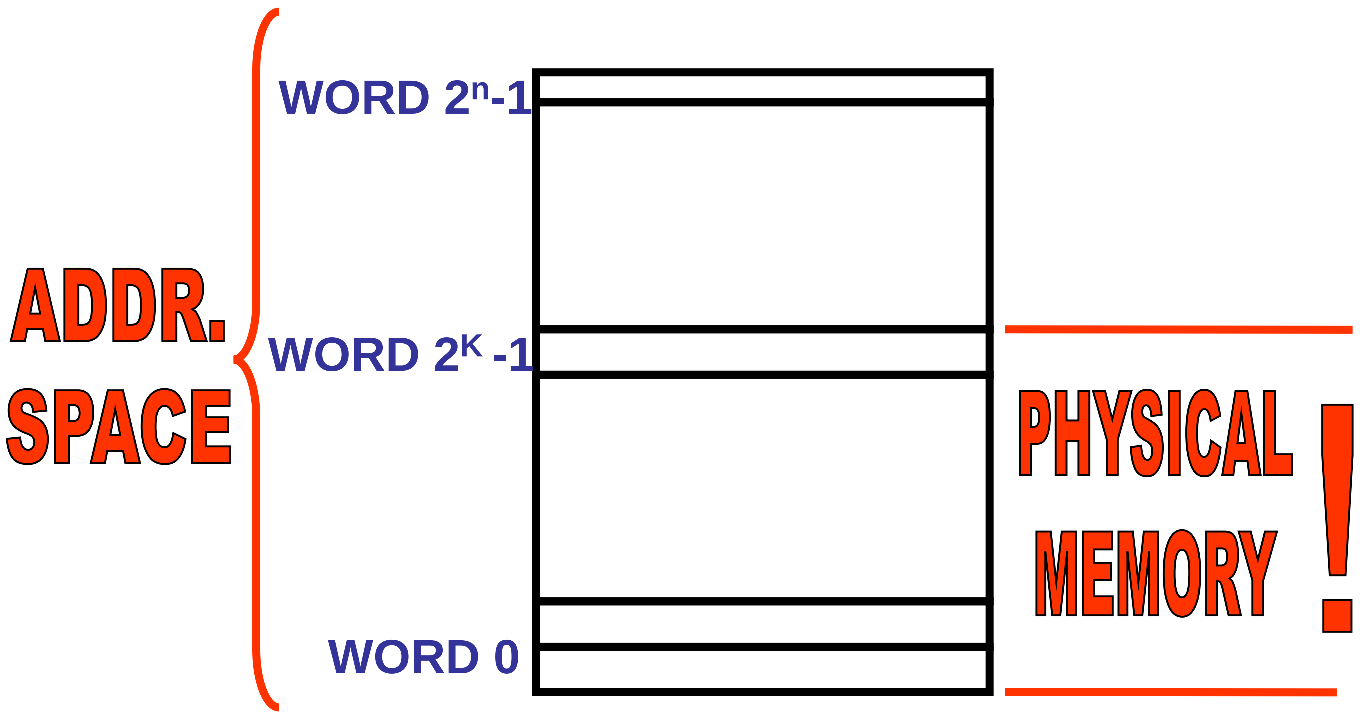
Refined Block Diagram



Basic Principles: Address Space

Physical (meaningful) addresses


CTRL
RD/~WR
MEMORY
ADDR
K
W
DATA IN/OUT


ADDR. SPACE
WORD 2ⁿ-1
WORD 2ᴷ -1
WORD 0
PHYSICAL MEMORY !