Computer Mathematics for Graduate Engineers

Week 1

Computer structure and organisation

KUAS 京都先端科学大学

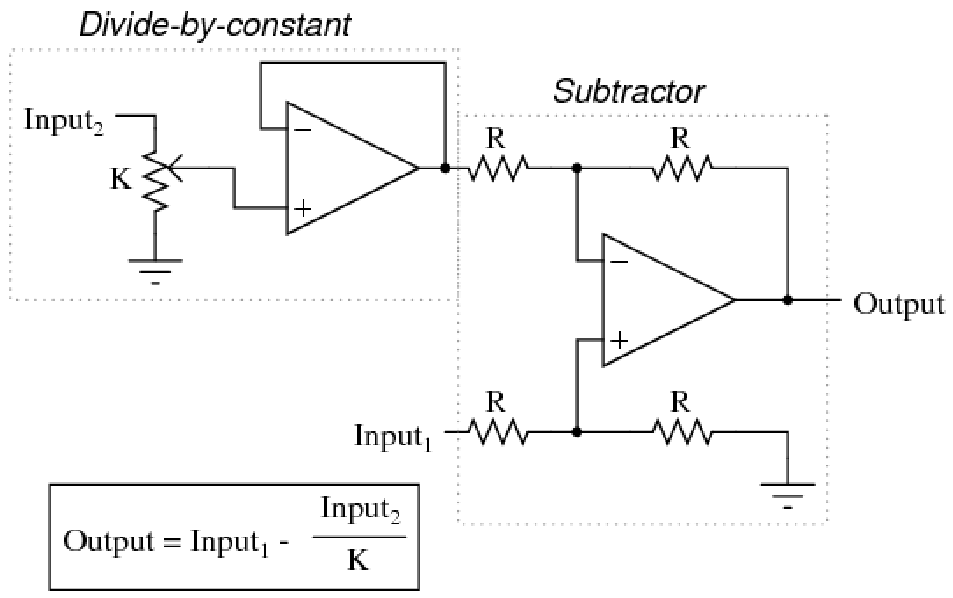
KYOTO UNIVERSITY of ADVANCED SCIENCE

Department of Mechanical and Electrical System Engineering

about this course

instructor:	(Prof. | Dr.) PIUMARTA
e-mail:	`ian.piumarta@kuas.ac.jp`
web:	`kuas.org/~piumarta`
location:	S401 (office), 4F Electronics Workshop (lab)
office hours:	send e-mail for appointment
course title:	Computer Mathematics for Graduate Engineers
course web site:	`kuas.org/~piumarta/cm`

what is a computer?



computer |kəm’pju:tə|

noun

an electronic device capable of receiving information (*data*) in a particular form and performing a sequence of operations in accordance with a predetermined but variable set of *instructions* to produce a result in the form of information or signals.

add, subtract, multiply, divide integrate, differentiate log, exp

- analogue voltages
- representing *continuous* values

what is a computer?

more recently:

- a *digital* system that stores, transforms, and communicates *data* in digital form

Manchester University *Small-Scale Experimental Machine*, 1948

data is represented by a fixed number of *discrete* (non-continuous) symbols

- called *digits*

data and instructions

data is any information that a computer can process

individual data values may represent

- numbers
- alphabetic characters
- any other encoded information (music, picture, video, etc.)

instructions are commands telling the computer how to manipulate data

- instructions are represented as numbers
- they are stored in the computer just like any other kind of data
- a sequence of instructions is a *program*
- programs can create, save, load, execute other programs
  - compilers and operating systems are programs that manipulate programs (which is easy, because programs are just data)

data representation

all data is stored and processed in the *binary* form

- that is, as a series of `0`s and `1`s
- e.g., `01010011000110111101`

bini (Latin, ‘two together’)
$\rightarrow$ *binarius*
$\rightarrow$ *binary* (English, ‘a pair’)

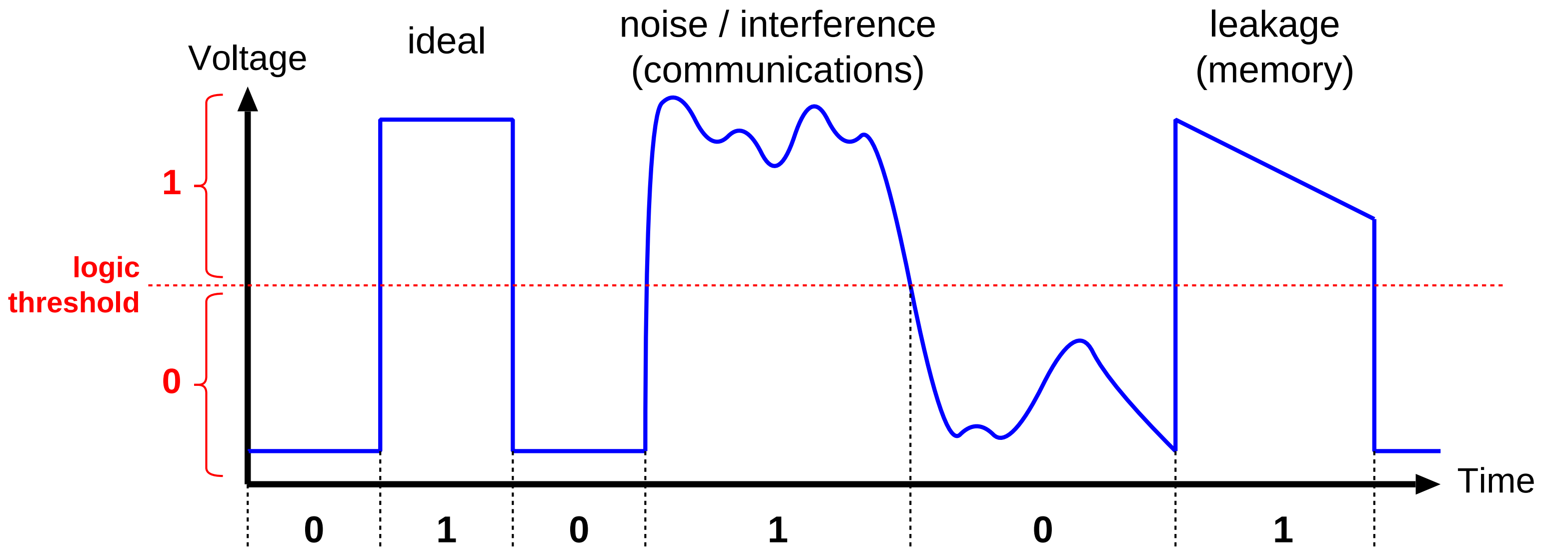
each binary digit is called a *bit*

- <u>**b**</u>inary <u>**i**</u>nformation digi<u>**t**</u>
- the smallest unit of information that can be transmitted, stored or processed

why binary?

advantages

- basic decision-making processes in logic are binary: “yes/no”, “true/false”, “`1/0`”
- hardware that manipulates binary information is easy to design
- binary signals are reliable, and easy to store as voltage levels in hardware

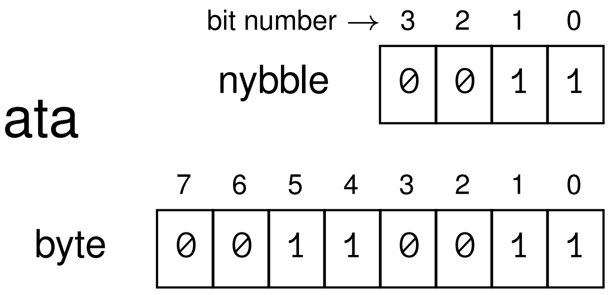


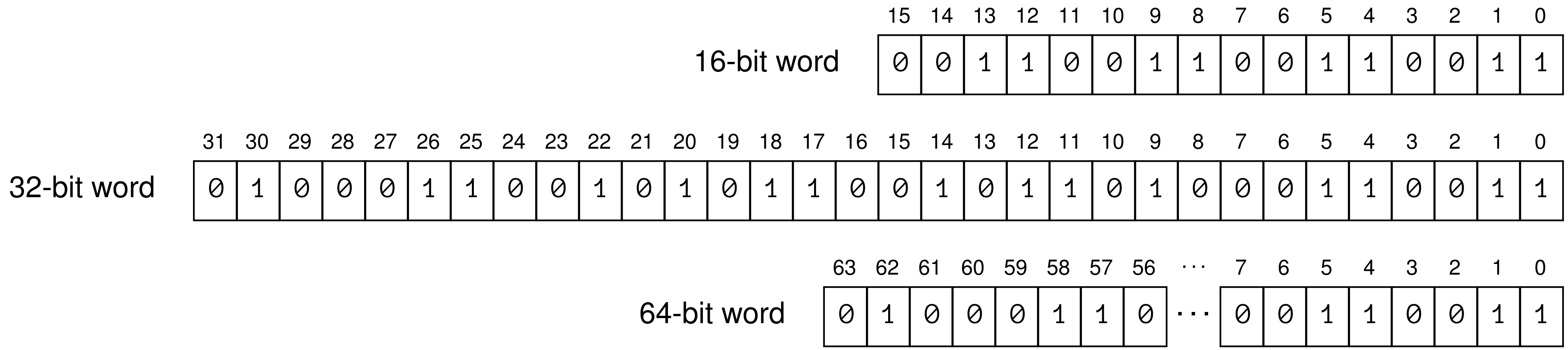
disadvantages

- we are more familiar with decimal numbers
- binary numbers are very long when written down

binary data units

bits are grouped into larger units to hold more meaningful data

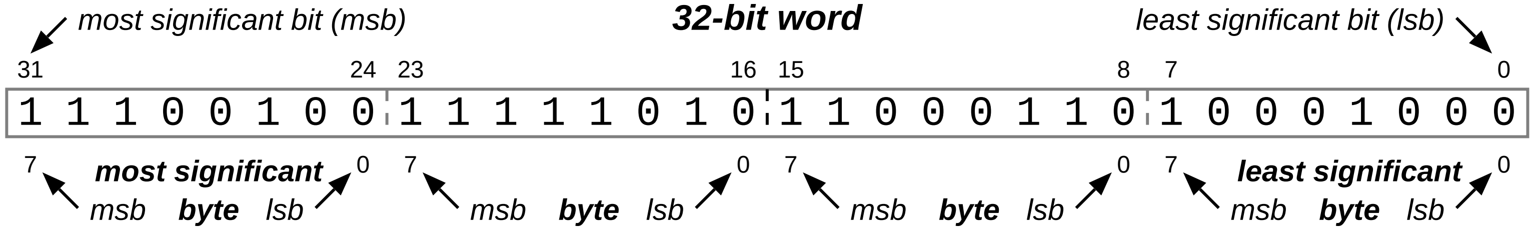




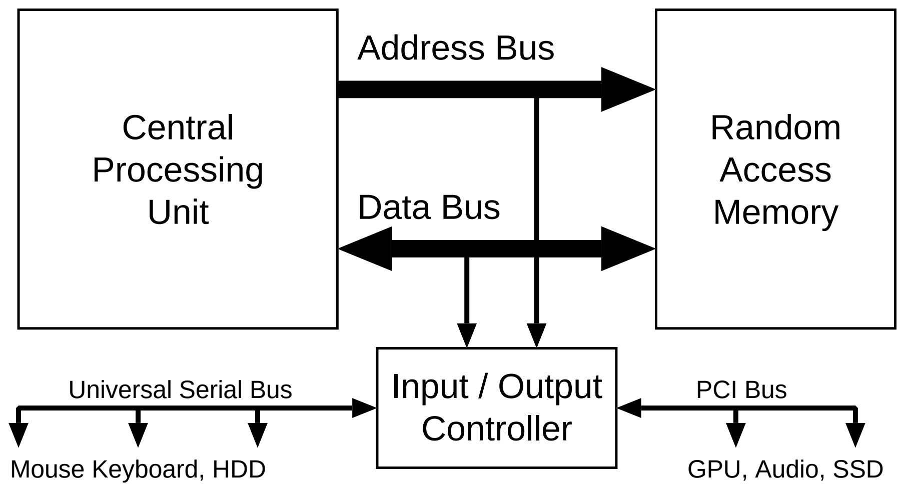
a *word* is usually the natural size of a datum (e.g., an integer in arithmetic)

the leftmost bit is the *most significant bit* (msb)

the rightmost bit is the *least significant bit* (lsb)



computer organisation



the *Random Access Memory* (RAM) is where data and program instructions are stored

- individual data bytes or words can be *read* and *written*
- access time is constant (data can be read/written in any order)

the *Central Processing Unit* (CPU) manipulates data according to program instructions

an I/O controller performs *input/output* operations (human interaction, data storage)

the *data bus* carries data between these units

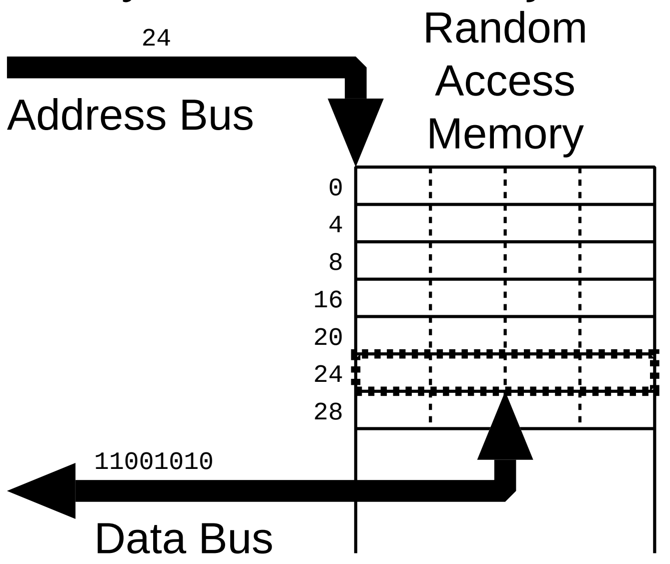
the *address bus* identifies which data item in memory to read or write

- or which input/output device should provide/receive data

main memory operation

each location in main memory has a unique *address*

- main memory locations are selected for reading or writing using their address
- memory addresses are just binary numbers



read: data_bus $\leftarrow$ memory[address_bus]

write: data_bus $\rightarrow$ memory[address_bus]

the address bus supplies addresses to the main memory

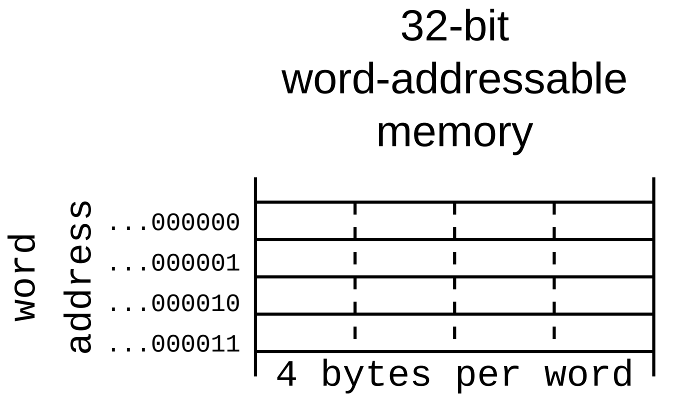
the data bus carries data between main memory and the CPU

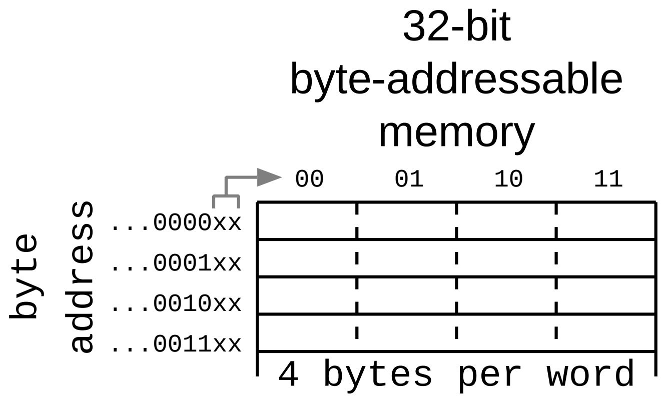
- write: copy the value on the data bus to memory at the address on the address bus
- read: copy the value in memory at the address on the address bus to the data bus

main memory organisation

word-addressable memory

- a memory address identifies an entire word
- consecutive words have consecutive addresses





byte-addressable memory, N bytes per word

- a memory address identifies a single byte
- consecutive words have addresses that increase by N

central processing unit

the Central Processing Unit (CPU), or ‘processor’, executes program instructions

most CPUs contain at least the following parts:

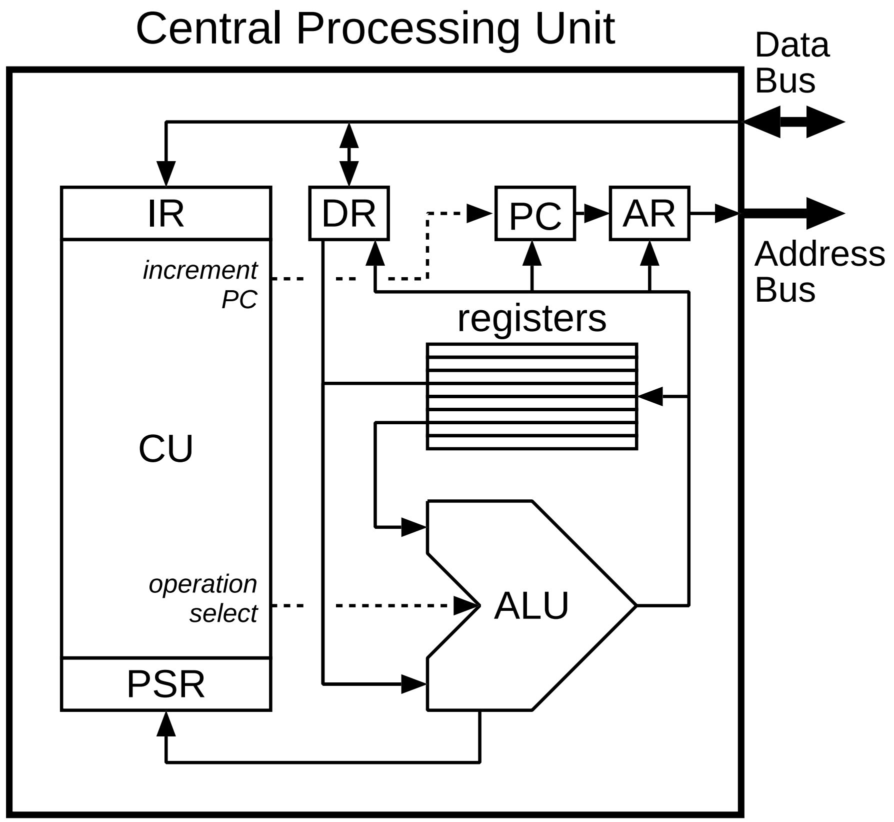
- **instruction register** (IR): holds the instruction currently being executed
- **control unit** (CU): controls the CPU’s execution of instructions
  - decodes the content of the instruction register
- **arithmetic and logic unit** (ALU): performs arithmetic/logical operations on data
  - addition, subtraction, multiplication, logic operations
- **general purpose registers** (GPRs): very fast word-sized memory locations
  - provide input data to the ALU, store output results from it
- **processor status register** (PSR): stores information about ALU results
  - zero, negative, overflow, carry, etc.
- **program counter** (PC): contains the address of the next instruction to execute

instruction execution

machine ('fetch-execute') cycle:
repeat

1. **fetch instruction**
   - copy PC to address bus
   - read from memory into IR
2. **decode instruction**
   - CU inspects the IR
3. **fetch operand(s)**
   - move data from memory into DR, if required
   - present DR and/or registers to ALU
4. **execute instruction**
   - ALU performs operation on data
5. **store result**
   - write ALU output into register or memory
6. **update PC**
   - with the address of the next insutruction

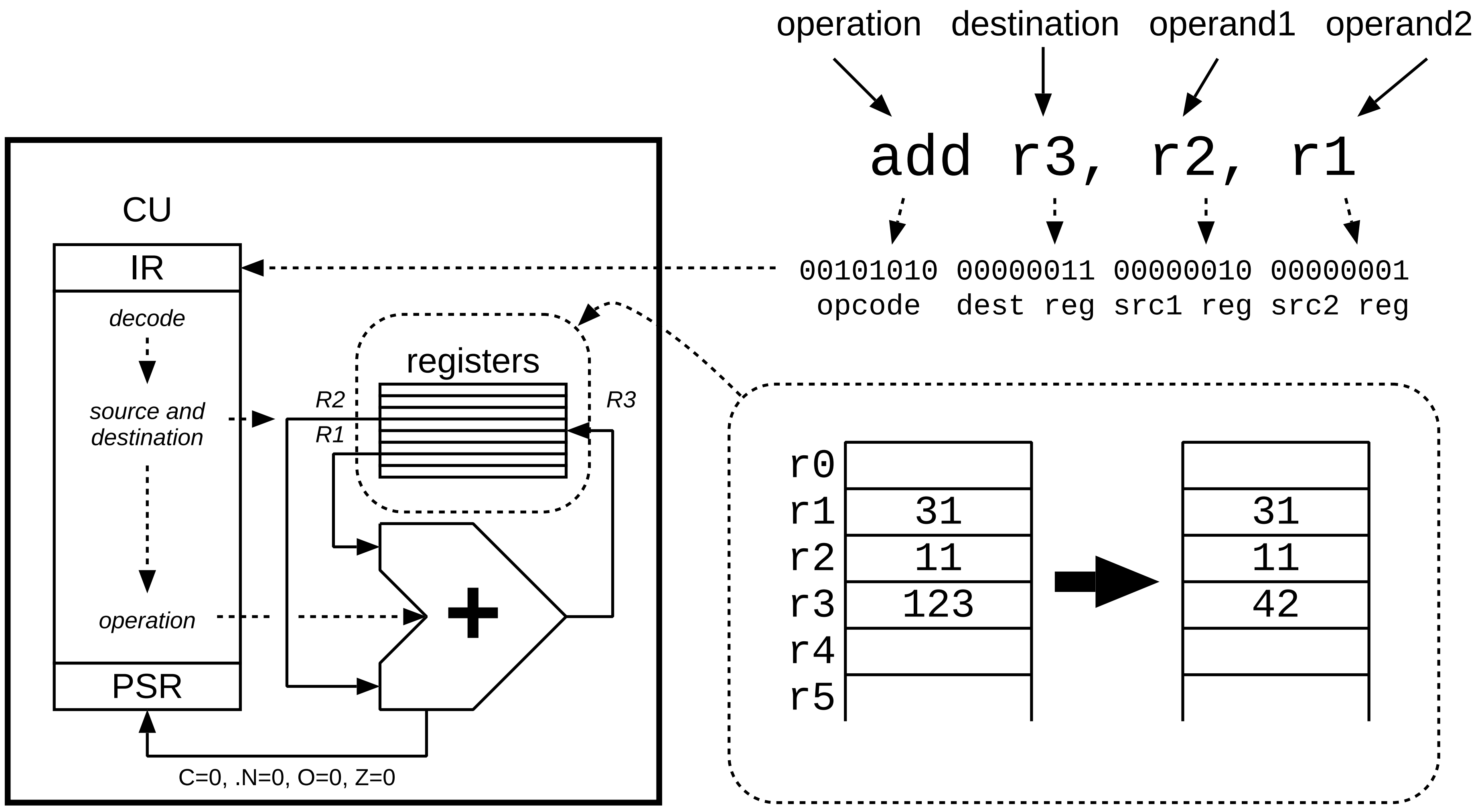
forever



DR: data register, holds data read from memory while ALU performs an operation
AR: address register, holds address for memory read/write operation

a typical machine instruction

add register 1 to register 2, putting the result in register 3



a typical machine program

search an array to find the index of a particular value, assuming that

- the variable `array` holds the address of the start of the array
- the variable `length` holds the length of the array
- the variable `value` holds the target value that we are searching for
- the index of the element should be returned in register `r0`
- if the target is not found, `-1` should be returned in `r0`

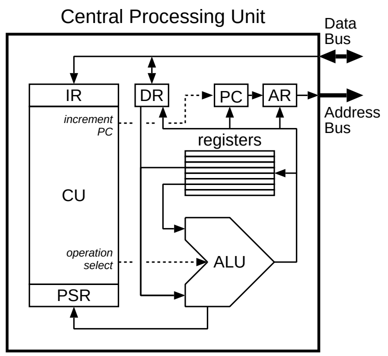
```
label    operation        operands      comment

         load             r1, array     load array address into register 1
         load             r2, length    load array length into register 2
         load             r3, value     load target value into register 3
         set              r0, 0         the next index to check is held in r0
next:    compare          r0, r2        check if we reached the end of the array
         jump_if_equal    fail          if so, target was not found
         load             r4, r1[r0]    fetch next element from array
         compare          r4, r3        compare it to the target value
         jump_if_equal    done          if found, return the corresponding index in r0
         add              r0, r0, 1     increment the index
         jump             next          continue searching at the next index
fail:    set              r0, -1        index -1 means 'target not found'
done:    halt                           r0 contains index of target element, or -1
```

input/output

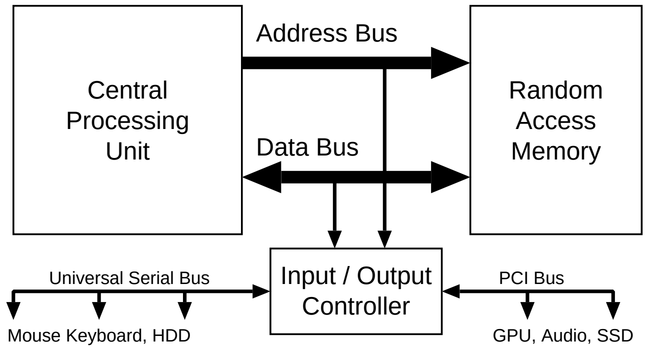
bus-based connections to *peripheral devices*

- *parallel bus*: complex connection, high speed, internal
  - transfers an entire byte or word at the same time
    - ∗ PCI (Peripheral Component Interconnect): graphics cards, network cards
    - ∗ SCSI (Small Computer System Interface): server disk drives
- *serial bus*: simple connection, lower speeds, internal/external
  - transfers one bit at a time
  - a byte (or word) has to be *serialized* for transmission
    - ∗ USB (Universal Serial Bus): keyboard, mouse, external memory
    - ∗ SATA (Serial AT Attachment): consumer disk drives

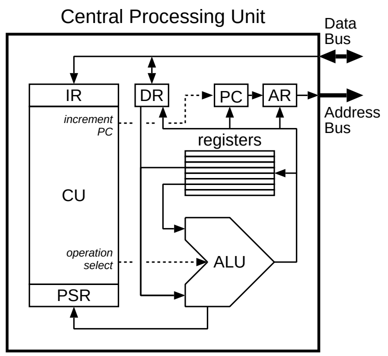
device	*direction*	*peer*	*typical bus*
keyboard, mouse	I	human	USB
printer	O	human	USB
graphics card	O	human	PCI
sound card	I/O	human	USB/PCI
HDD, SSD	I/O	computer	USB/PCI/SATA/SCSI
optical drive	I/O	computer	USB/SATA
network card	I/O	computer	PCI



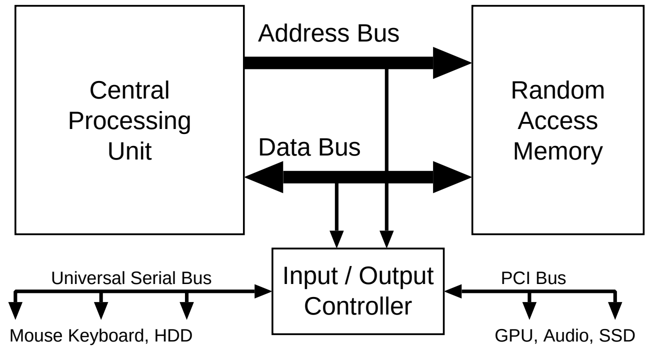
course outline



- what components does a computer need to store and process data?
- how are numbers represented inside the computer?
  how are numbers represented outside the computer (on paper)?
  how does arithmetic work in binary?
- how is data protected against damage (e.g., noise, degradation)?
  how are large amounts of data stored/transmitted efficiently?
- how is the ALU designed?
  how can its mathematical operations be implemented as simple logic operations?
  how are registers and memory built from simple logic functions?
  how does the control unit provide sequencing using only logic?
- how is the control unit designed and described, mathematically?
  how how does that relate to other sequential things, like language?
- how are compuer languages analysed and translated so that we can program the CPU?



course outline



1 **Overview of computing systems**

Digital and binary systems. Data, instructions, memory, busses, input/output. CPU, ALU, control.

2 – 4 **Mathematics of data**

Number systems. Representations: binary, octal, hexadecimal. Radix conversion. Negative numbers. Binary arithmetic.

5 – 6 **Mathematics of information**

Basic information theory. Error detection and correction. Data compression.

7 – 9 **Mathematics of computer hardware**

Boolean logic and algebra. Digital circuits, ALU design. Combinational and synchronous logic. Data storage, memory cells.

10 – 12 **Mathematics of control**

Models of computation. Sequential logic, control unit design. Finite State Machines. Determinism and non-determinism.

13 – 15 **Mathematics of language**

FSMs as regular expressions and regular grammars. Grammar types. Linear vs. recursive languages. Parsing, ambiguity. Translation and interpretation of progamming languages.

glossary

address — a number that uniquely identifies a byte or word of data in memory (or a peripheral device controller) during a read or write operation.

address bus — a parallel bus that carries address information to memory (or a peripheral device controller).

address register — a CPU register that holds the address being accessed during a read or write operation.

arithmetic and logic unit (ALU) — the part of the CPU that performs arithmetic and logical operations on one or two operands.

binary — any system that distinguishes between two values, states, etc.

bit — a single binary information digit, the smallest possible unit of data, usually written as ‘`0`’ or ‘`1`’.

bus — a serial or parallel connection over which information (data or addresses) is communicated.

byte — a group of eight bits.

central processing unit (CPU) — the part of a computer that executes programs and manipulates data.

continuous — a value that can change smoothly over a certain range (temperature, wind speed, etc.), rather than being discrete.

glossary

data — values that represent information or instructions within a computer.

data bus — a bus that carries data from one part of a computer (or peripheral device) to another.

data register — a CPU register that holds data after it is read from memory, or while it is being written to memory.

register — a fast memory location implemented within the CPU holding data that must be accessed frequently or quickly.

digital — any system that uses discrete symbols to represent data.

digit — a symbol representing information such as a logical or numerical value.

discrete — a quantity that takes on specific predefined values (integers, days of the week, etc.), rather than varying continuously.

input — a device or action that moves data into the computer.

input/output controller — hardware that controls a peripheral device.

instruction — a binary word that can be interpreted by the control unit as a command for manipulating data within the CPU.

instruction register (IR) — a register that holds the instruction currently being executed.

glossary

least significant bit — the rightmost bit in a binary value, having the lowest bit number and the least numerical significance.

least significant byte — the rightmost byte in a binary value, having the lowest numerical significance.

machine (fetch-execute) cycle — the (infinitely-repeating) set of steps performed by the control unit to ensure that an instruction is executed.

most significant bit — the leftmost bit in a binary value, having the highest bit number and the most numerical significance.

most significant byte — the leftmost byte in a binary value, having the most numerical significance.

nybble — a group of four bits.

octet — another word for ‘byte’.

operand — an input value to an arithmetic or logical operation.

output — a device or action that moves data out from the computer.

parallel — a multi-wire connection that can move an entire byte or word in a single transfer.

parallel bus — a bus that uses multiple wires to connect all relevant bits between two devices, allowing entire bytes or words to be moved in a single transfer.

glossary

peripheral device — an electronic machine connected to a computer, typically performing an input/output or storage function.

processor status register — a register that stores information about the results of ALU operations (negative, zero, overflow, carry, etc.).

program — a sequence of instructions that are executed by the CPU to manipulate data.

program counter — a register that holds the address of the next instruction to be executed.

random access memory (RAM) — a large, relatively fast memory holding most of the data being processed, that can be accessed in any order without penalty.

read — an operation that moves data from memory (or a peripheral device) to the CPU.

serial — a single-wire connection that moves one bit at a time, requiring N transfers to move a N-bit byte or word.

serial bus — a bus that uses a single wire to transfer one bit of information at a time between two devices.

serialization — converting a parallel quantity (such as a register or memory location) into a sequence of single bits that can be transmitted serially.

word — a unit of data representing a natural unit of processing (e.g., the size of an integer manipulated by the ALU, or the size of the address bus, etc.) for a CPU.

write — an operation that moves data from the CPU to memory (or a peripheral device).