EP1200 Introduction to Computing Systems Engineering

Computer Architecture

Perspective on Machine and Assembly Languages

- Hack is a simple machine language
- Designed for the simple Hack computer architecture
  - Few registers
  - One data type (integer)
  - Multiplication and division in software

- Cumbersome programming – no space for a memory address within an instruction

- User friendly syntax: D=D+A instead of ADD D,D,A

Perspective on Machine and Assembly Languages

Machine languages classification

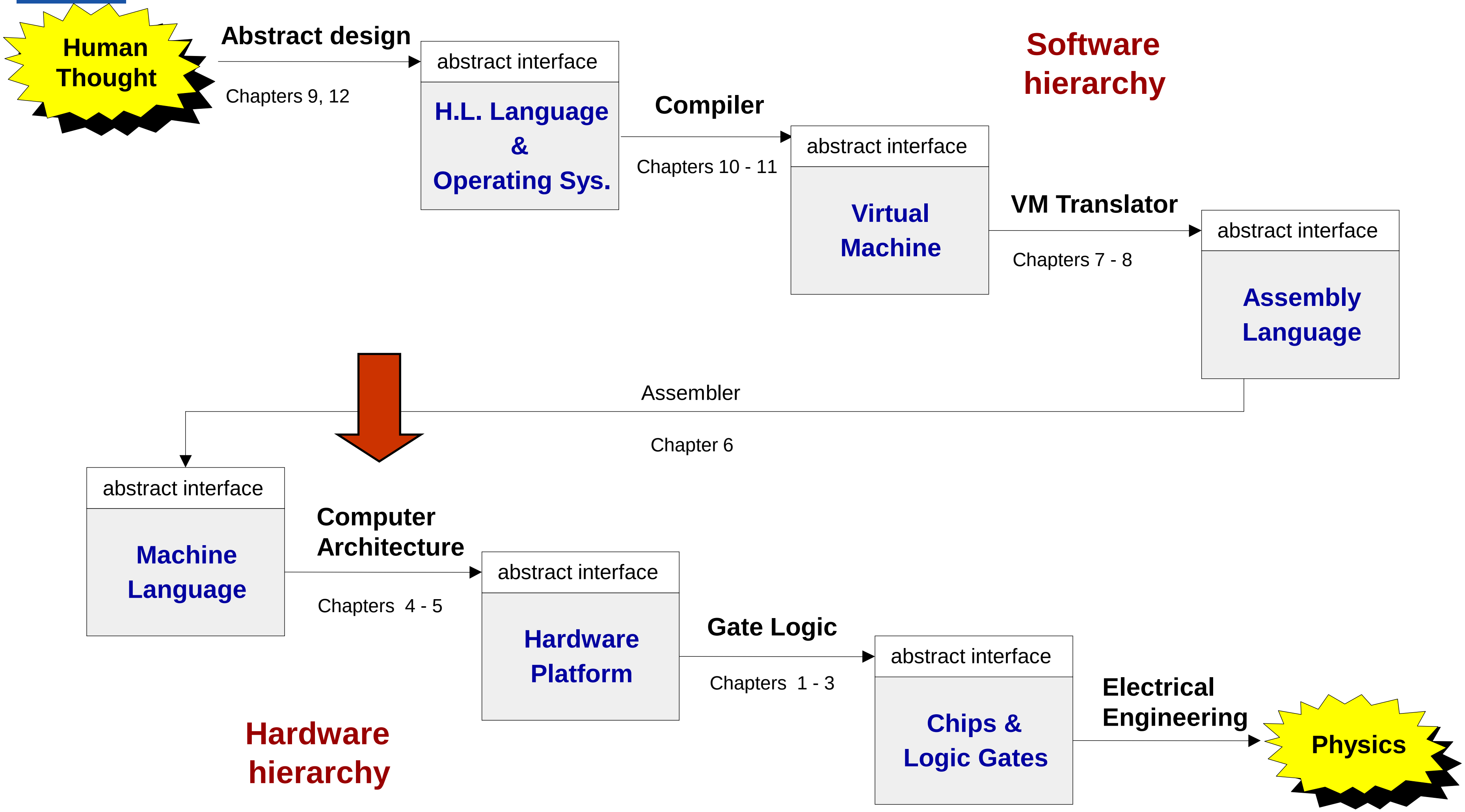
- Traditionally: complex hardware or long code
  - CICS – complex instruction set computer (Intel, AMD)
    - little memory requirement, complicated CPU HW
  - RISC – reduced instruction set (mobile systems, game consoles but even supercomputers)
    - Long code, emerged when memory became cheap and fast
- New variants:
  - MICS – minimal instruction set (for embedded systems, special purpose processors)
  - VLIW – very long instruction word (for parallel processing)

Perspective on Machine and Assembly Languages

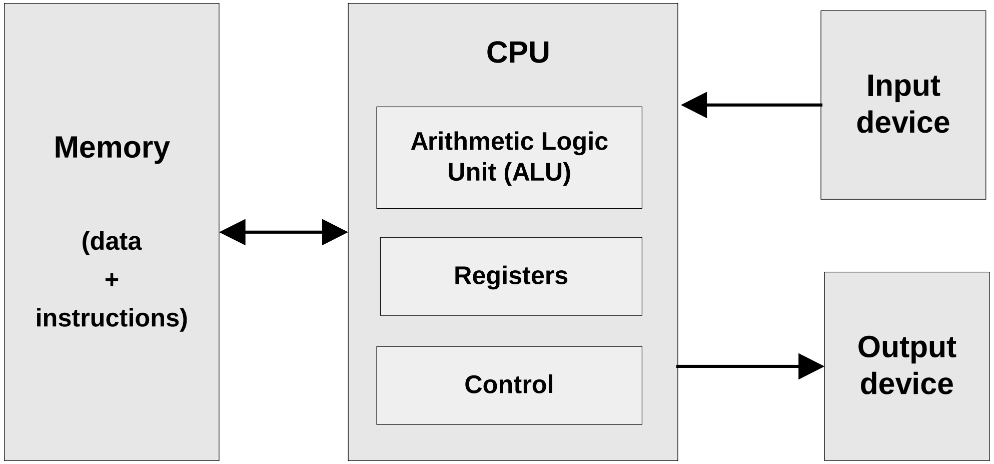
Some typical differences

- Memory operations per instruction
  - CICS – allows “load and store”
  - RICS – only one memory operation per instruction “ load or store”
- Operands per instruction
  - ½ address machine: Hack
  - 1 address machine: one operand ….
  - 3 address machines
    - E.g. add a,b,c (c=a+b, in one instruction)

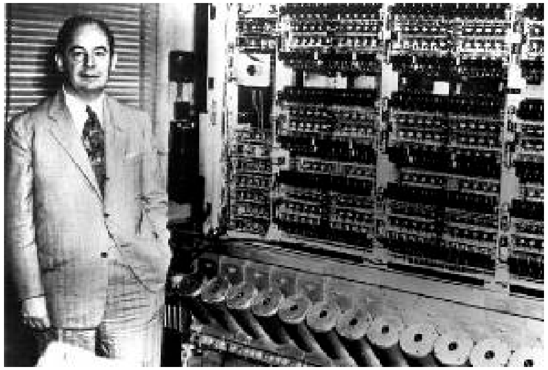
Where we are at:

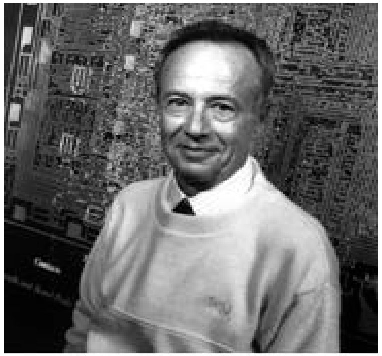


Computer architecture – Von Neumann machine (circa 1940)



- Stored program concept: the program code is stored in memory and can be manipulated, just like data





Andy Grove (Intel) (and others) made it small and fast

John Von Neumann (and others) made it possible

The Hack chip-set and hardware platform

Elementary logic gates

- Nand
- Not
- And
- Or
- Xor
- Mux
- Dmux
- Not16
- And16
- Or16
- Mux16
- Etc.

done

Combinational chips

- HalfAdder
- FullAdder
- Add16
- Inc16
- ALU

done

Sequential chips

- DFF
- Bit
- Register
- RAM8
- RAM64
- RAM512
- RAM4K
- RAM16K
- PC

done

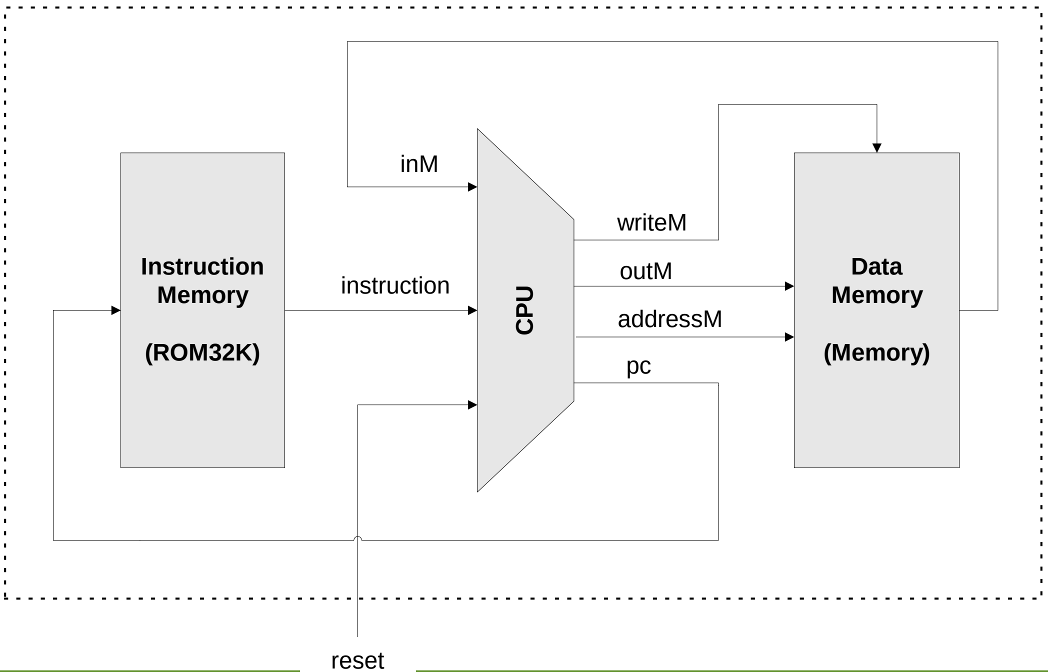
Computer Architecture

- Memory
- CPU
- Computer

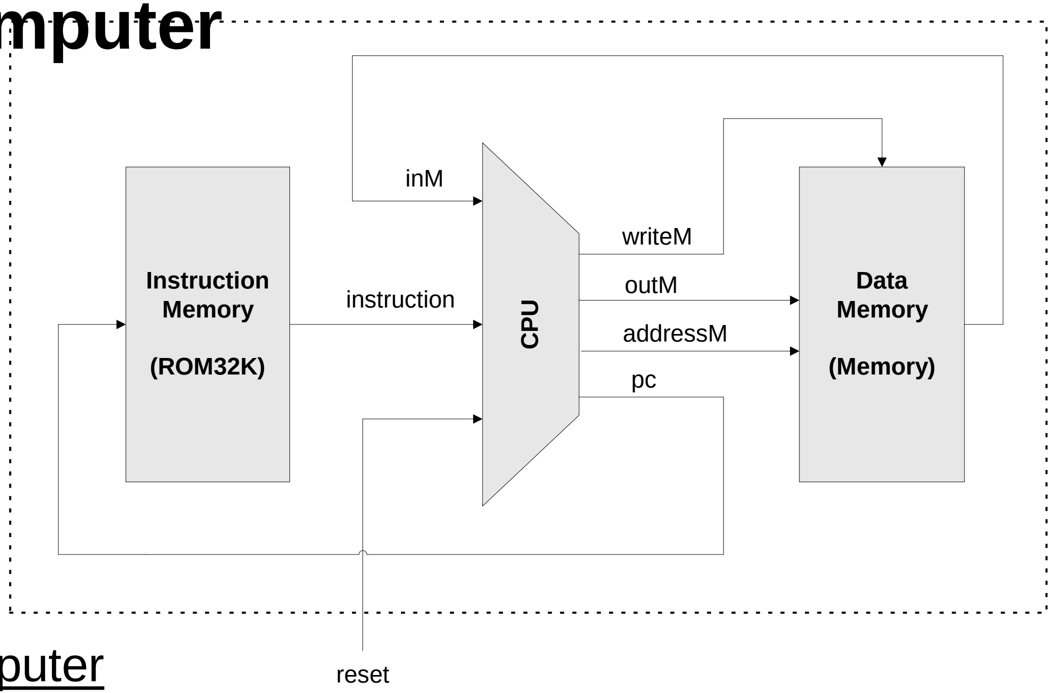
this lecture

The Hack computer

- A 16-bit Von Neumann platform (actually more Harvard platform: instruction and data memory separated)
- Designed to execute programs written in the Hack machine language – the architecture and the language needs to be designed together!



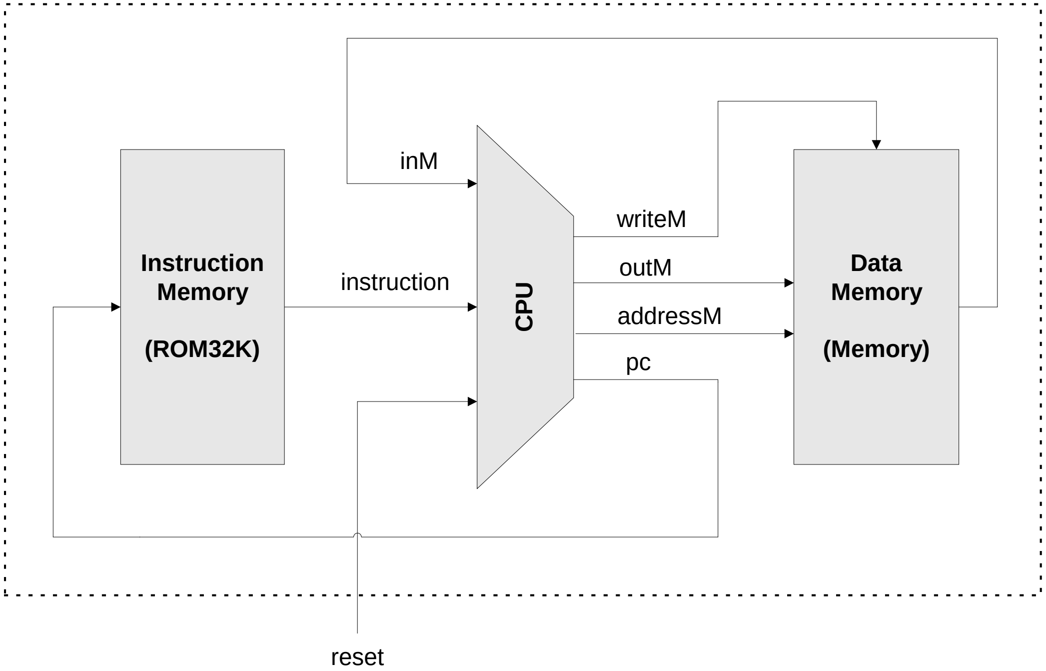
The Hack computer



Main parts of the Hack computer

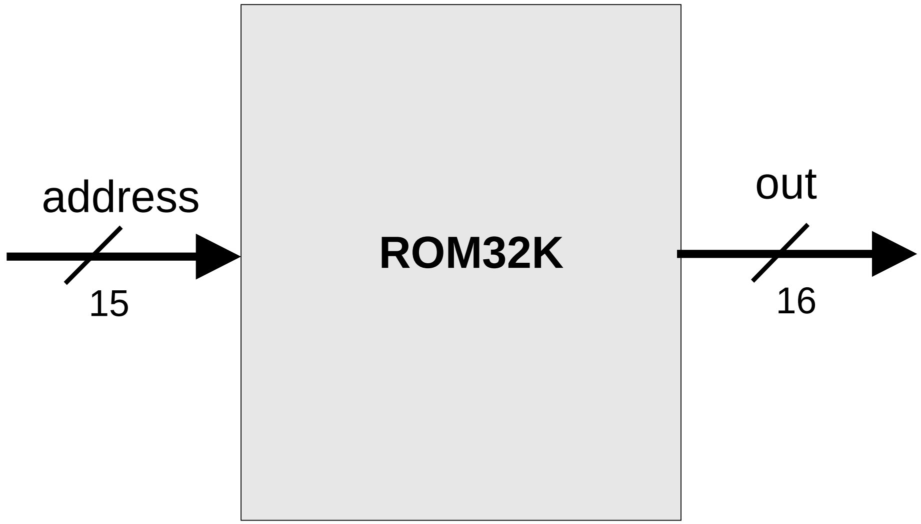
- ❑ Instruction memory (ROM) – we consider it given
- ❑ Data memory: Memory (RAM), Screen (memory map), Keyboard (memory map)
- ❑ Central Processing Unit, CPU
- ❑ Computer (the logic that holds everything together)

The Hack computer – implementation



- ROM, RAM, Memory
- Wiring as on the block diagram – implementation is simple

Instruction memory - ROM

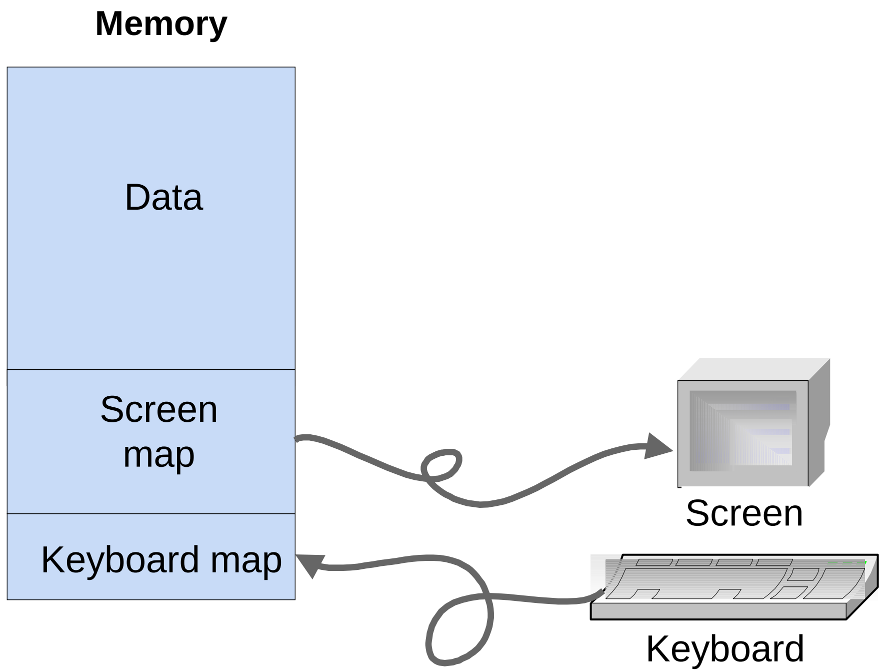


Function:

- The ROM is pre-loaded with a program written in the Hack machine language (note, you can not dynamically change the program!)
- The ROM chip always emits a 16-bit number, this number is interpreted as the current instruction.
- We consider the ROM given, you do not need to implement it.

Memory (RAM, Screen, Keyboard)

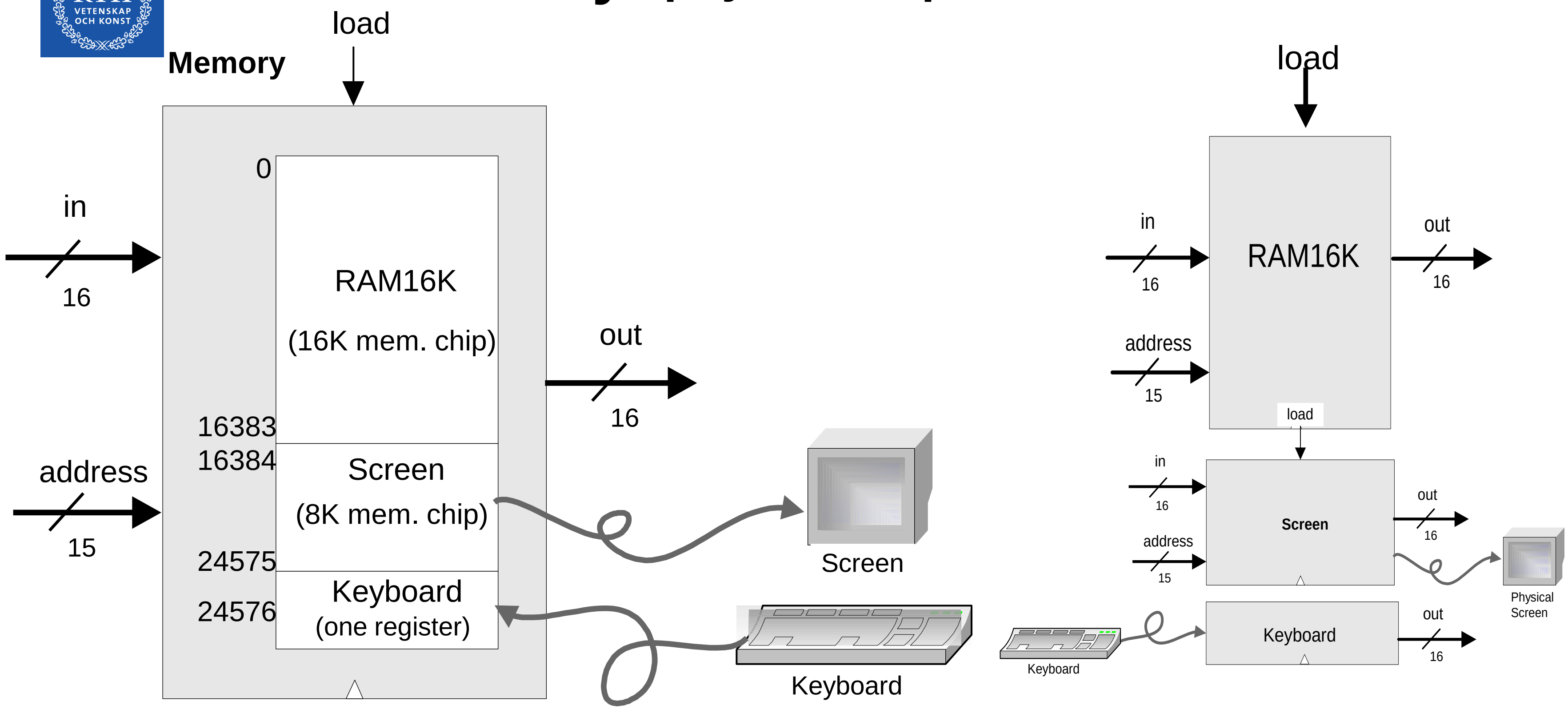
conceptual / programmer's view



Using the memory:

- ❑ To record or recall values (e.g. variables, objects, arrays), use the first 16K words of the memory
- ❑ To write to the screen (or read the screen), use the next 8K words of the memory
- ❑ To read which key is currently pressed, use the next word of the memory.

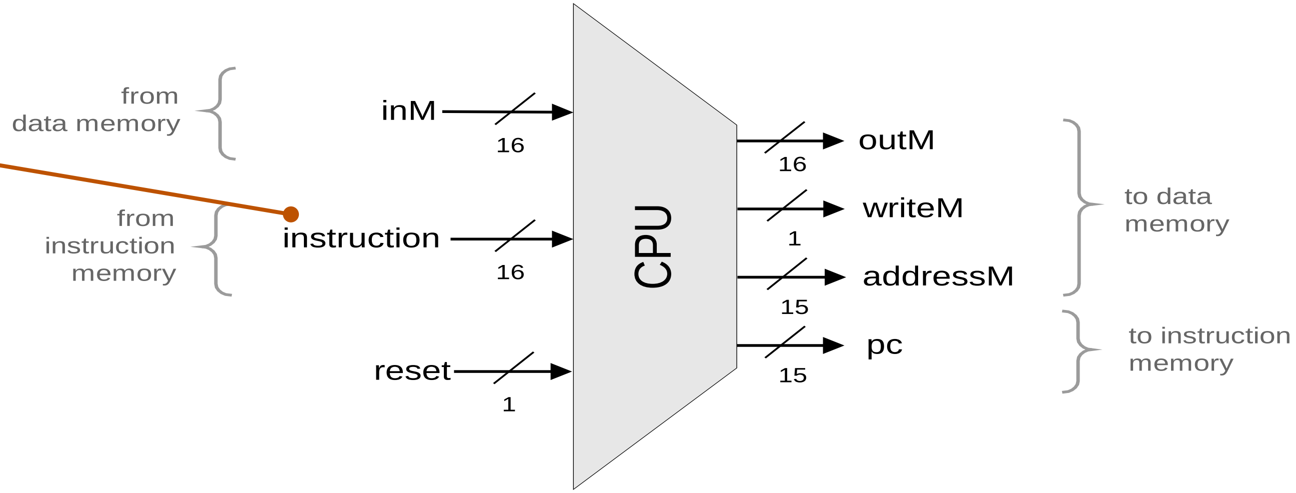
Memory: physical implementation



- The Memory chip: integrates the three chip-parts RAM16K, Screen, and Keyboard into a single address space.
- Implementation challenge: addressing

CPU – ALU, A, D registers, PC

A Hack machine language instruction stated as a 16-bit value



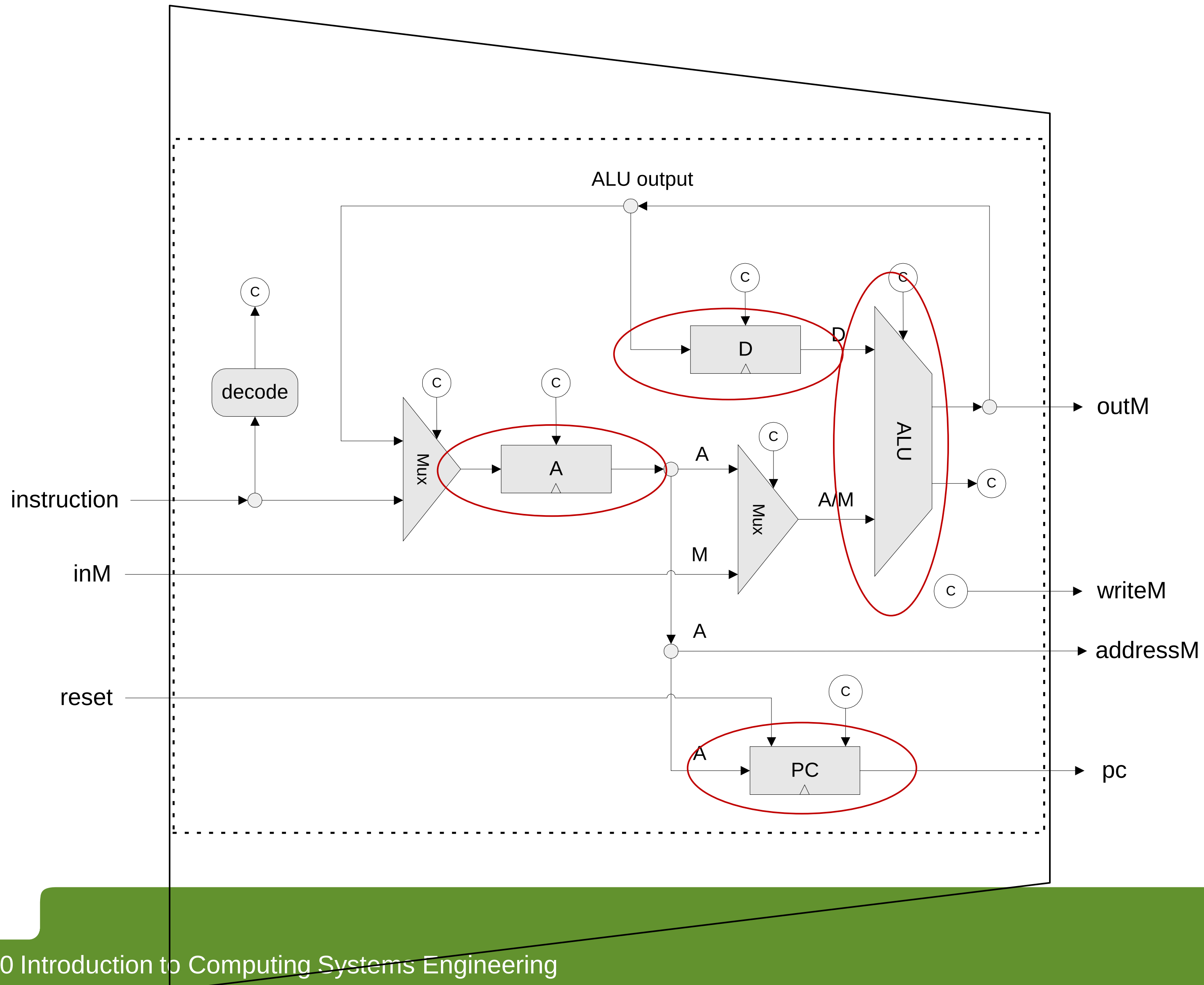
CPU elements

- ALU (arithmetic unit)
- A (address), D (data) registers and PC (program counter)

CPU operation

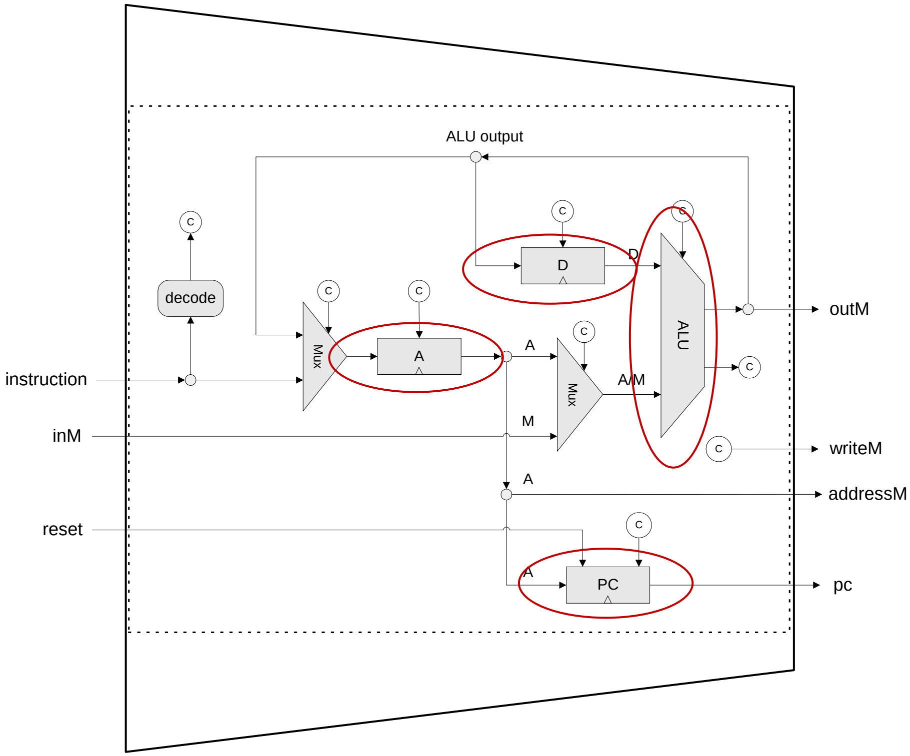
- Executes HACK machine language instruction
- Reads from and writes to memory
- Reads or sets A, D and PC registers (inside the CPU)

CPU (ALU, registers A, D, PC) implementation

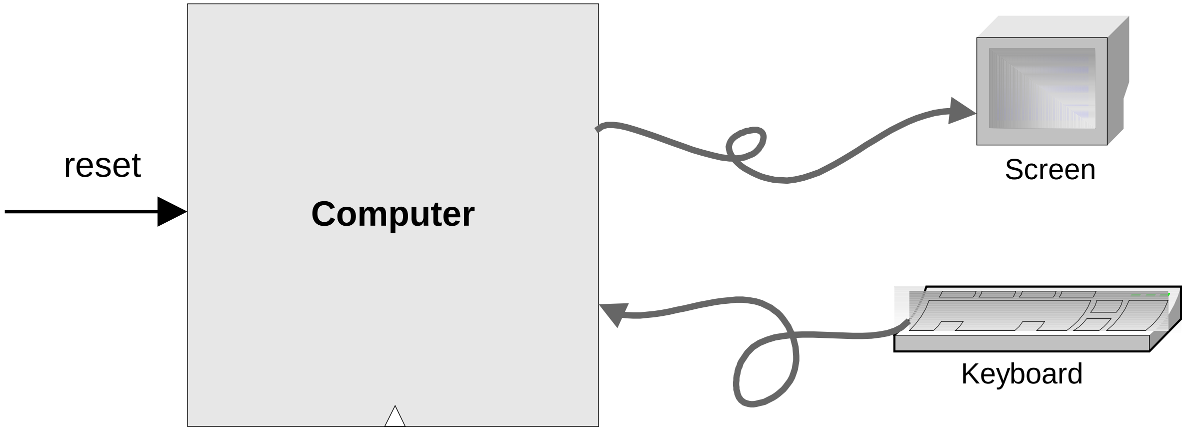


CPU (ALU, registers A, D, PC) implementation

- Implementation is challenging
  - ALU functions
  - Machine Language instructions
- Break it to small problems
  - Instruction decoding
  - D register wiring
  - A register wiring
  - ALU: inputs and outputs
  - Jumps and Program counter



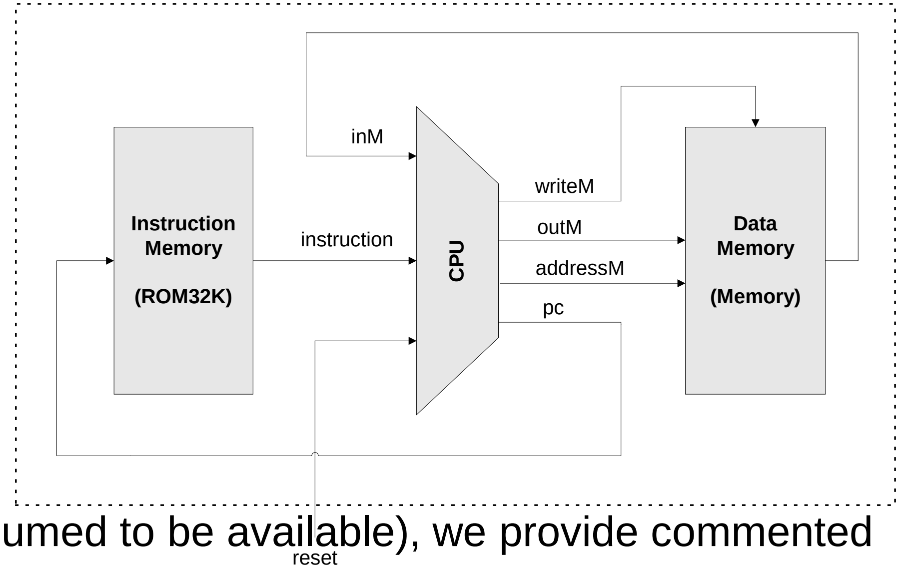
Project - Computer-on-a-chip




```
Chip Name:  Computer  // Topmost chip in the Hack platform
Input:      reset
Function:   When reset is 0, the program stored in the
            computer's ROM executes. When reset is 1, the
            execution of the program restarts. Thus, to start a
            program's execution, reset must be pushed "up" (1)
            and "down" (0).

            From this point onward the user is at the mercy of
            the software. In particular, depending on the
            program's code, the screen may show some output and
            the user may be able to interact with the computer
            via the keyboard.
```

Project - Computer-on-a-chip implementation (Chapter 5 project)



- ROM given
- Implement Memory (RAM, Screen, Keyboard) in the hardware simulator
- Implement CPU, (A-register, D-register assumed to be available), we provide commented template
- Implement Computer from CPU, ROM and Memory
- Test Memory and CPU in separation, then test the Computer (test programs are available)
- Always comment your code!

Project - Computer-on-a-chip implementation

- Reading and tools
  - *Repeat Chapter 2 ALU*
  - *Repeat Chapter 4 Machine Language basic concepts*
  - Chapter 5 Computer architecture
  - Tools: Hardware simulator (the one used in Projects 1-2)
- Hand in project by April 4, 8.00.