

Computer Architecture

Shimon Schocken

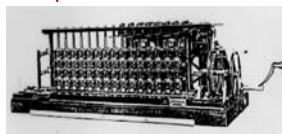
Spring 2005

Some early computers



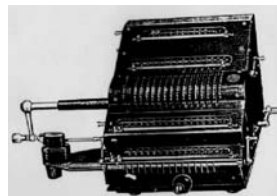
**Blaise Pascal,
1623-1662**

Multiplier

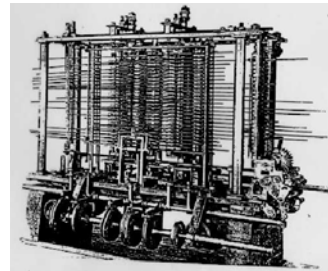


**Gottfried Leibniz,
1646-1716**

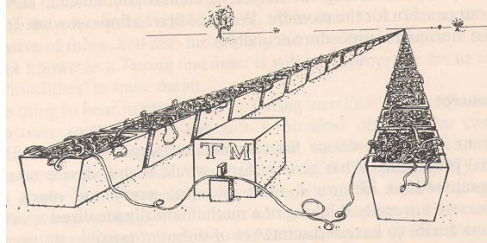
Divider



Logarithm



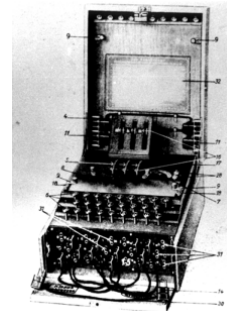
Turing machine (c. 1930)



Alan Turing
1912 - 1954

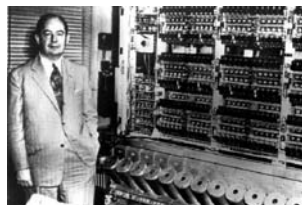
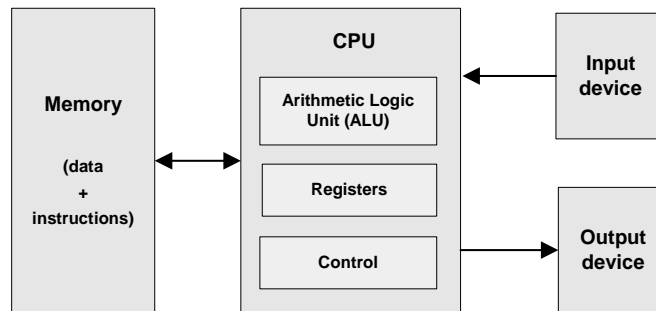
Informal description:

- A *tape*, divided into *cells*, each containing a symbol.
- A *head* that can read and write symbols and move left and right.
- A *state register* that stores the machine's state.
- An *action table* that tells the machine what symbol to write, how to move the head, and what its new state will be, given the symbol it has just read on the tape and the current state.



The Enigma

Von Neumann machine (c. 1940)



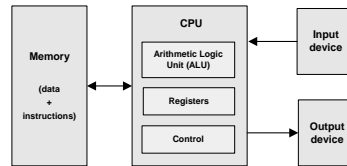
John Von Neumann ... made it possible



Andy Grove ... made it small (and fast).

Von Neumann machine

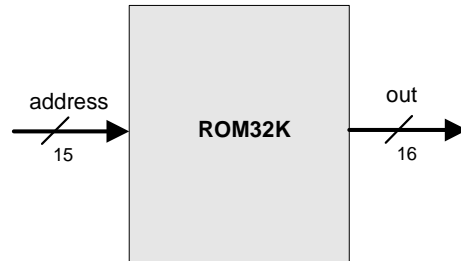
- Memory = data + instructions
- The memory is addressable
- CPU = ALU, data registers, address registers, control
- Fetch-execute cycle: the execution of the current instruction involves one or more of the following micro tasks:
 - have the ALU compute some value
 - manipulate internal registers
 - read a word from the memory
 - write a word to the memory
 - In the process of executing these tasks, figure out which instruction to fetch and execute next
- Input / output devices.



The Hack computer

- 16-bit Von Neumann platform
- *Instruction memory* and *data memory* are two physically separate units
- I/O: 512 by 256 black and white screen, regular keyboard
- Designed to execute programs written in the Hack machine language
- Can be easily built from the chip-set that we built so far in the course
- Main parts:
 - Instruction memory
 - Memory:
 - Data memory
 - Screen
 - Keyboard
 - CPU
 - Computer

Instruction memory

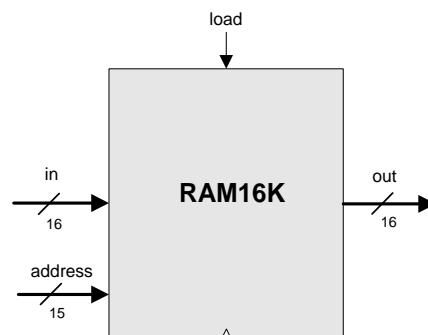


Function:

- Pre-loaded with a machine language program
- Always emits the current instruction:

`out = ROM32K[address]`

Data memory



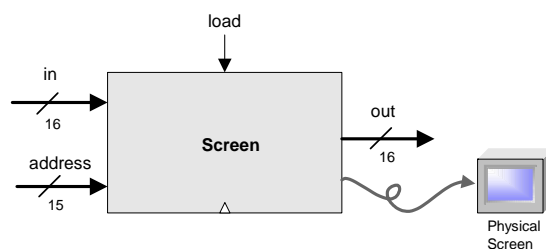
Accessing the memory

- Low level: Set `address`, `in`, `load`, etc.
- High level: Use `peek(address)` and `poke(address, value)` (OS services)

Lecture plan

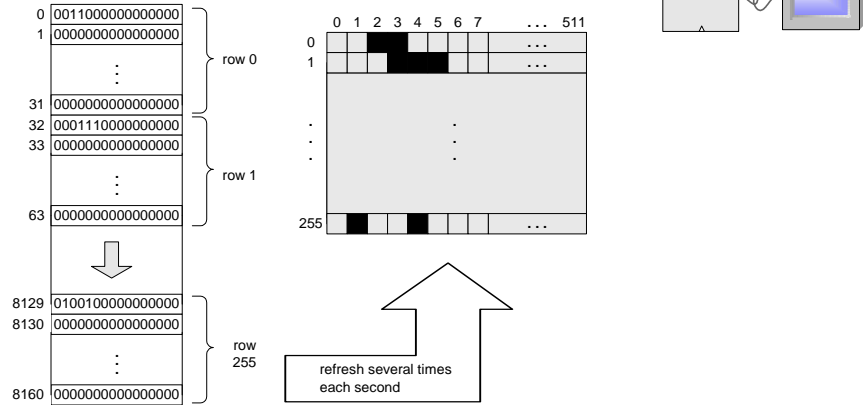
- ✓ ■ Instruction memory
- Memory:
 - ✓ ● Data memory
 - Screen
 - Keyboard
- CPU
- Computer

Screen



- Functions exactly like a 16-bit 8K RAM :
 - `out = Screen[address]`
 - If `load` then `Screen[address] = in`
- Side effect: continuously refreshes a 256 by 512 black-and-white screen.

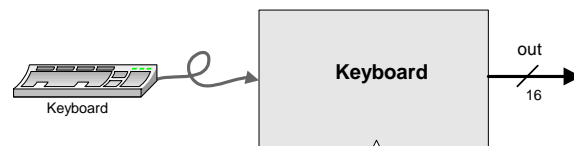
Screen



Writing pixel (x,y) to the screen:

- Low level: Set the $y\%16$ bit of the word found at `Screen[x*32+y/16]`
- High level: Use `drawPixel(x,y)` (OS service)

Keyboard



- Side effect: connected to a physical keyboard
- Function: out = the ASCII code of the pressed key, or 0 if no key is pressed.

Key pressed	Keyboard output	Key pressed	Keyboard output
newline	128	end	135
backspace	129	page up	136
left arrow	130	page down	137
up arrow	131	insert	138
right arrow	132	delete	139
down arrow	133	esc	140
home	134	f1-f12	141-152

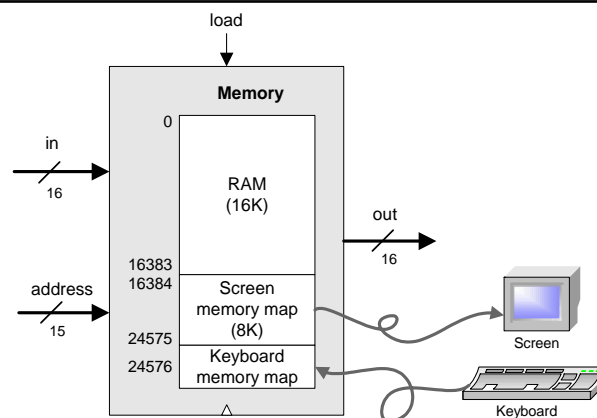
Reading the keyboard:

- Low level: probe the contents of the `Keyboard` register
- High level: Use `keyPressed()` (OS service)

The Hack computer

- ✓ ■ Instruction memory
- Memory:
 - ✓ ● Data memory
 - ✓ ● Screen
 - ✓ ● Keyboard
- CPU
- Computer

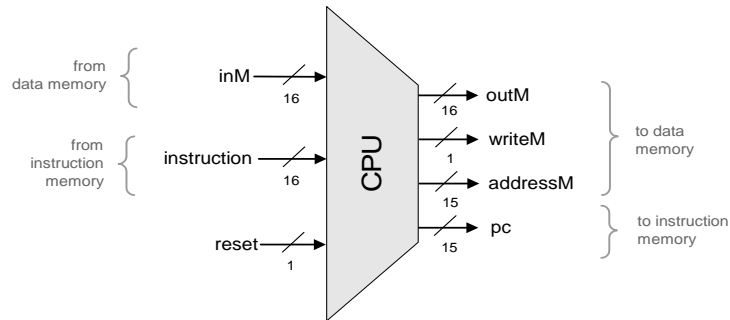
Memory



Function:

- Access to any address > 24576 is invalid
- Access to any address in the range 16,384 – 24,575 results in accessing the screen memory map
- Access to address 24,576 results in accessing the keyboard memory map.

CPU

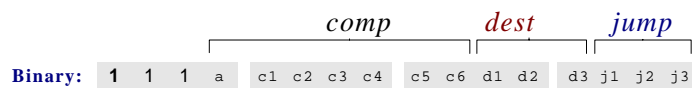


Function:

- Executes the *instruction* according to the Hack machine language specification. The D and A in the language specification refer to CPU-resident registers, while M refers to Memory[A]. The *inM* input holds the value of this location.
- If the *instruction* includes a directive to write a value to M, the value is placed in *outM*, the address is placed in *addressM*, and *writeM* is asserted.
- If *reset*=1, then *pc* is set to 0; Otherwise, *pc* is set to the address resulting from executing the current instruction.

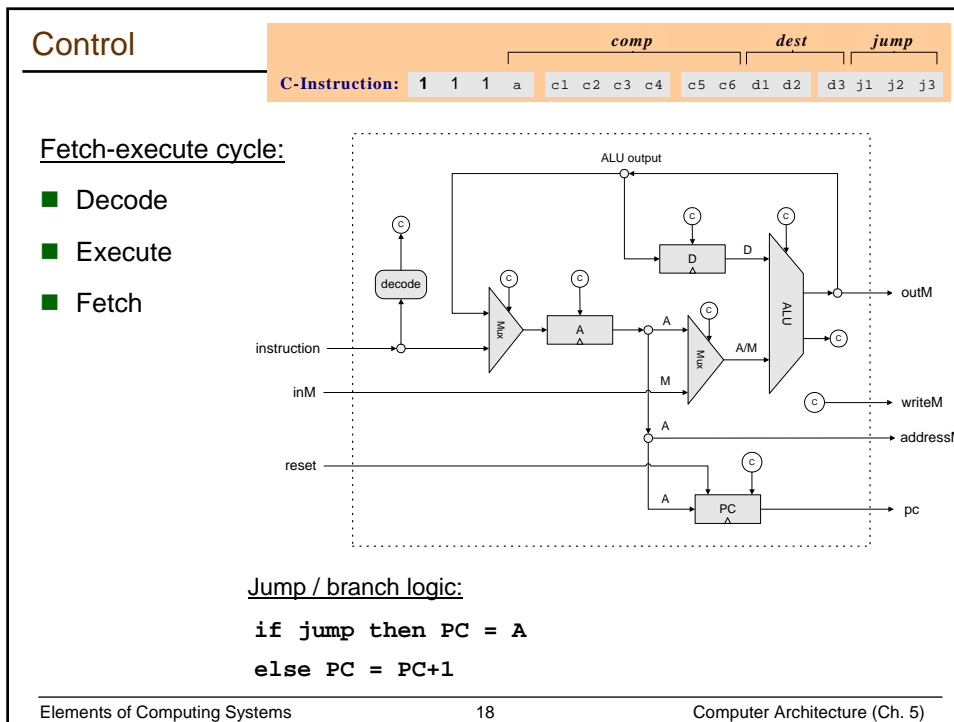
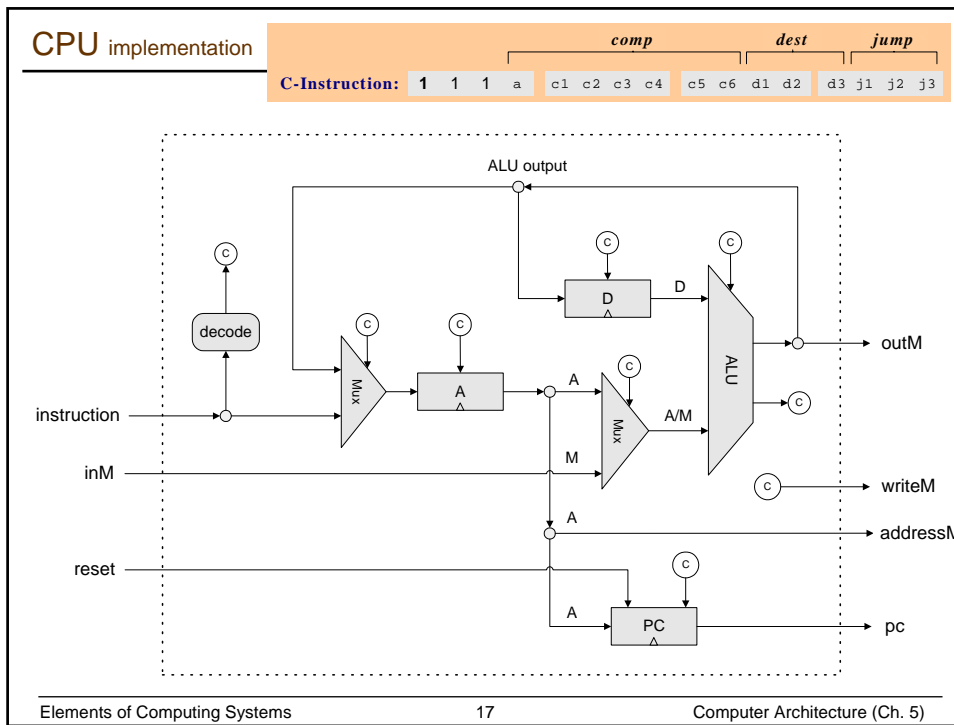
Let's re-visit the C-instruction ...

Symbolic: *dest=comp; jump*



(when a=0) <i>comp</i>	c1	c2	c3	c4	c5	c6	(when a=1) <i>comp</i>	d1	d2	d3	Mnemonic	Destination (where to store the computed value)
0	1	0	1	0	1	0		0	0	0	null	The value is not stored anywhere
1	1	1	1	1	1	1		0	0	1	M	Memory[A] (memory register addressed by A)
-1	1	1	1	0	1	0		0	1	0	D	D register
D	0	0	1	1	0	0		0	1	1	MD	Memory[A] and D register
A	1	1	0	0	0	0	M	1	0	0	A	A register
!D	0	0	1	1	0	1		1	0	1	AM	A register and Memory[A]
!A	1	1	0	0	0	1	!M	1	1	0	AD	A register and D register
-D	0	0	1	1	1	1		1	1	1	AMD	A register, Memory[A], and D register
-A	1	1	0	0	1	1	-M					
D+1	0	1	1	1	1	1						
A+1	1	1	0	1	1	1	M+1					
D-1	0	0	1	1	1	0						
A-1	1	1	0	0	1	0	M-1					
D+A	0	0	0	0	1	0	D+M					
D-A	0	1	0	0	1	1	D-M					
A-D	0	0	0	1	1	1	M-D					
D&A	0	0	0	0	0	0	D&M					
D A	0	1	0	1	0	1	D M					

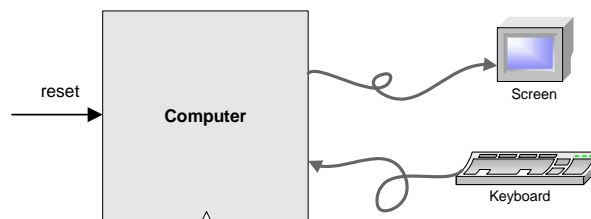
j1 (out < 0)	j2 (out = 0)	j3 (out > 0)	Mnemonic	Effect
0	0	0	null	No jump
0	0	1	JGT	If out > 0 jump
0	1	0	JEQ	If out = 0 jump
0	1	1	JGE	If out ≥ 0 jump
1	0	0	JLT	If out < 0 jump
1	0	1	JNE	If out ≠ 0 jump
1	1	0	JLE	If out ≤ 0 jump
1	1	1	JMP	Jump



The Hack computer

- ✓ ■ Instruction memory
- ✓ ■ Memory:
 - ✓ • Data memory
 - ✓ • Screen
 - ✓ • Keyboard
- ✓ ■ CPU
- Computer

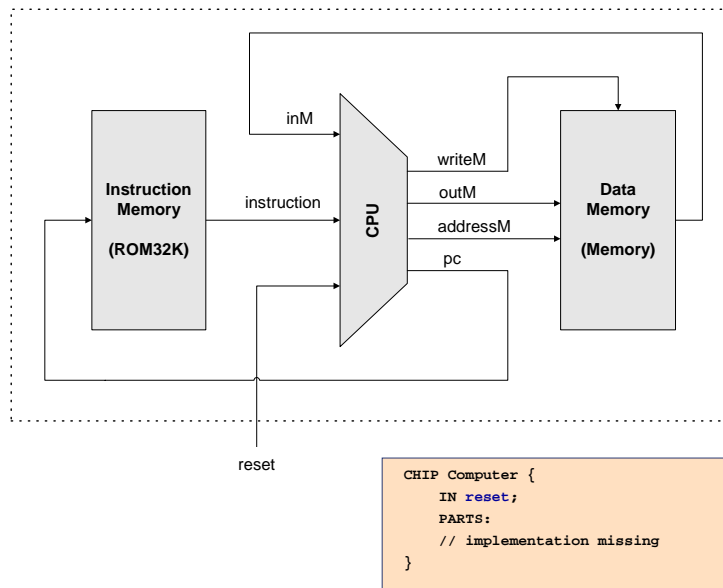
Computer



```
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.
```

Computer



Related topics

- More memory maps
- Special boards / drivers
- Clock / MH
- Efficiency
- CISC / RISC (HW/SW trade-off)
- Laptops
- Dedicated / embedded computers
- And more ...