CRICOS PROVIDER 00123M

School of Computer Science

COMP SCI 2000 Computer Systems
Lecture 8

Review – Last week

- We introduced the HACK machine.
- Assembly language commands make it easier to program the machine.
- There are lots of machine language commands
  - These will be hardware specific so you have to know about the platform. (HACK, in this case.)

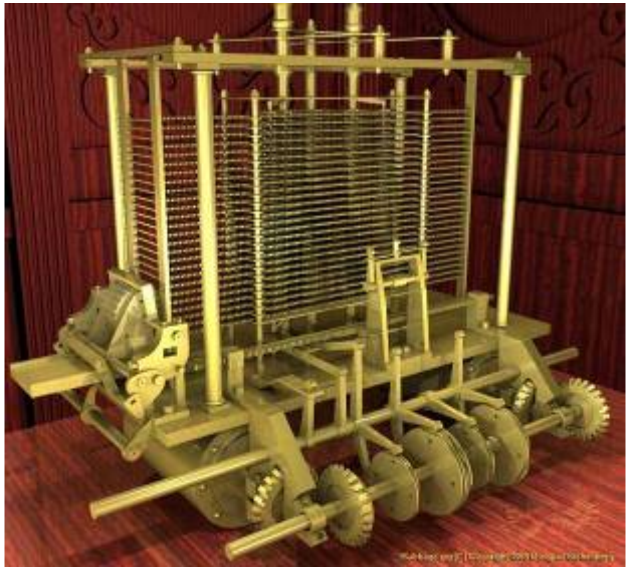
What we're doing now

- This week we're going to talk about:
  – History of Architecture
  – Memory and I/O
  – The HACK machine
  – The CPU and basic computers

Babbage's Analytical Engine (1835)

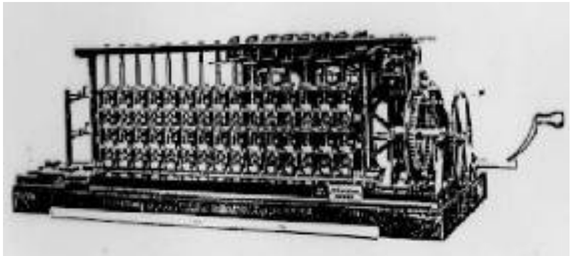
"We may say most aptly that the Analytical Engine weaves algebraic patterns just as the Jacquard-loom weaves flowers and leaves"
(Ada Lovelace, the first programmer.)

Charles Babbage (1791-1871)

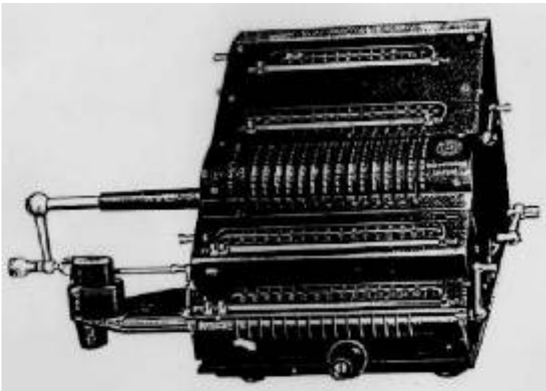


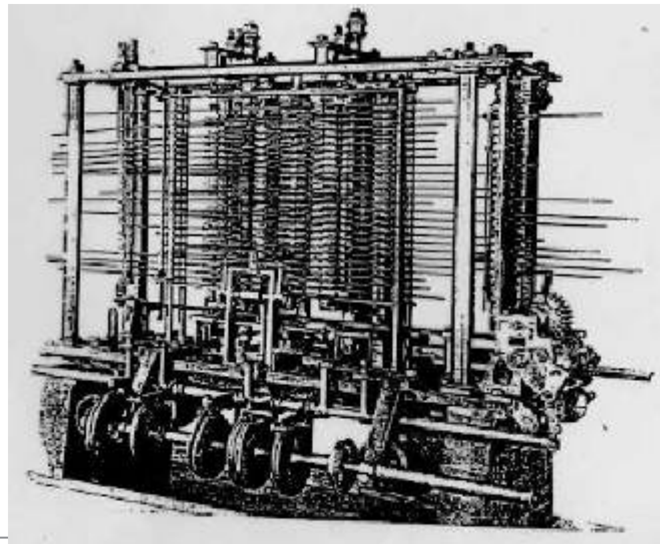
Some early computers and computer scientists

Blaise Pascal
1623-1662



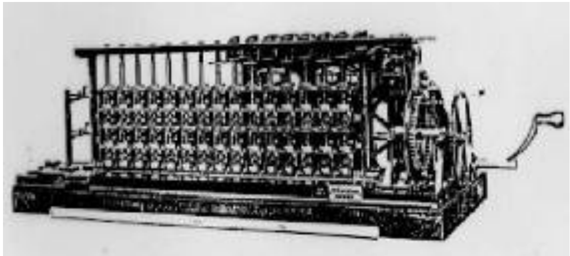
Gottfried Leibniz
1646-1716



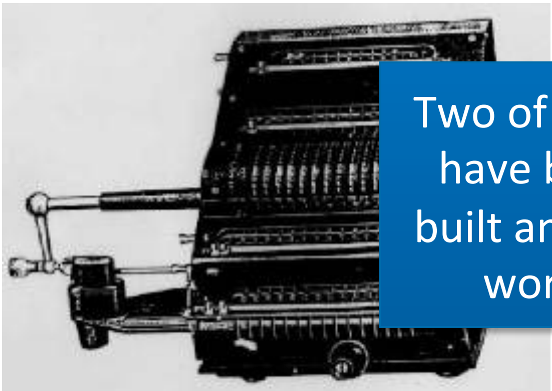


Some early computers and computer scientists

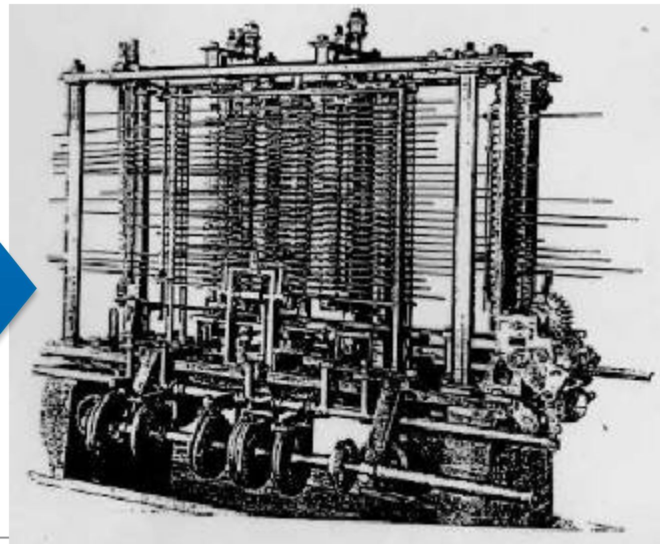
Blaise Pascal
1623-1662



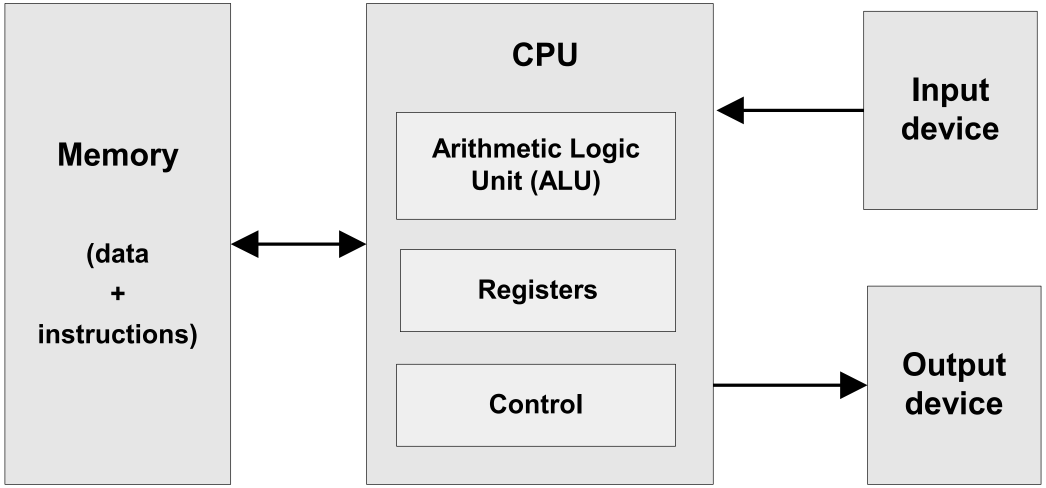
Gottfried Leibniz
1646-1716

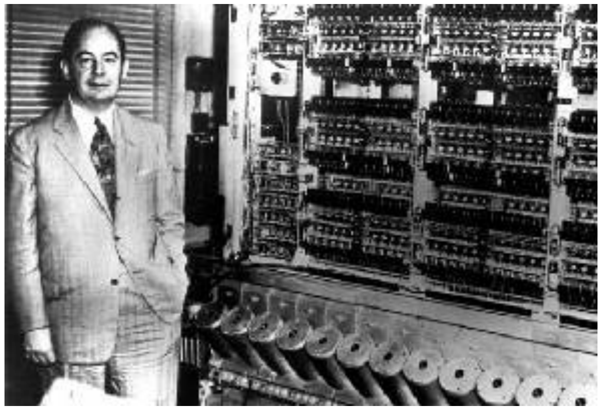


Two of these have been built and still work!



Von Neumann machine (circa 1940)





Stored
program
concept!

John Von Neumann (and others) ... made it possible

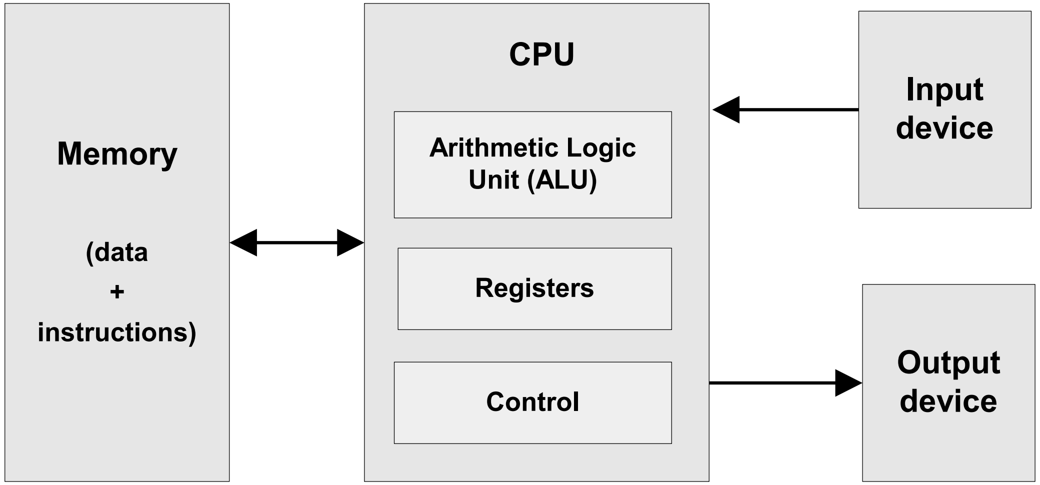
Andy Grove (and others) ... made it small and fast.

Turing?

- Hey, what about Turing and the Bombes?
- Do you know why we're not mentioning him here?

Worksheet question 1

Processing logic: fetch-execute cycle



Executing the *current instruction* involves one or more of the following micro-tasks:

- Have the ALU compute some function $\text{out} = f\,(\text{register values})$
- Write the ALU output to selected registers
- As a side-effect of this computation, figure out which instruction to fetch and execute next.

What's in the HACK chip-set?

- NAND…
  - And?

The Hack chip-set and hardware platform

Elementary logic gates

- Nand
- Not
- And
- Or
- Xor
- Mux
- Dmux
- Not16
- And16
- Or16
- Mux16
- Or8Way
- Mux4Way16
- Mux8Way16
- DMux4Way
- DMux8Way

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

The Hack chip-set and hardware platform

Elementary logic gates

- Nand
- Not
- And
- Or
- Xor
- Mux
- Dmux
- Not16
- And16
- Or16
- Mux16
- Or8Way
- Mux4Way16
- Mux8Way16
- DMux4Way
- DMux8Way

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 week

What we're doing now

- This week we're going to talk about:
  - Programming languages
    - Quick overview
    - History
    - High level languages can be … ear identical) across different hardware platforms.
  - Machine languages
    - Manipulating memory using a processor and a set of registers
    - Will be different across different hardware platforms
  - Machine language in HACK

We did this on Monday!

The Hack computer

- A 16-bit Von Neumann platform
- The *instruction memory* and the *data memory* are physically separate
- Screen: 512 rows by 256 columns, black and white
- Keyboard: standard
- 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 of the Hack computer:

- Instruction memory (ROM)
- Memory (RAM):
  - Data memory
  - Screen (memory map)
  - Keyboard (memory map)
- CPU
- Computer (the logic that holds everything together).

Lecture / construction plan

- Instruction memory
- Memory:
  - Data memory
  - Screen
  - Keyboard
- CPU
- Computer

Instruction memory

What does this look like from previous experience?

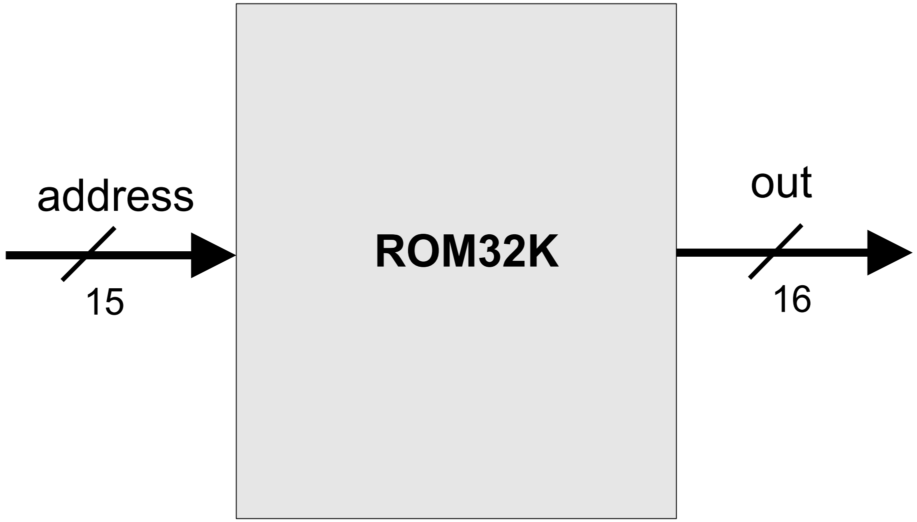
<u>Function:</u>

- The `ROM` is pre-loaded with a program written in the Hack machine language
- The `ROM` chip always emits a 16-bit number:

```
out = ROM32K[address]
```

- This number is interpreted as the *current instruction*.

Instruction memory



Function:

- The `ROM` is pre-loaded with a program written in the Hack machine language
- The `ROM` chip always emits a 16-bit number:

```
out = ROM32K[address]
```

- This number is interpreted as the *current instruction*.

Data memory

<u>Low-level (hardware) read/write logic:</u>

To read `RAM[k]`: set `address` to k,
probe `out`

To write `RAM[k]=x`: set `address` to k,
set `in` to x,
set `load` to `1`,
run the clock

What does this look like from previous experience?

<u>High-level (OS) read/write logic:</u>

To read `RAM[k]`: use the OS command `out = peek(k)`

To write `RAM[k]=x`: use the OS command `poke(k,x)`

`peek` and `poke` are OS commands whose implementation should effect the same behavior as the low-level commands

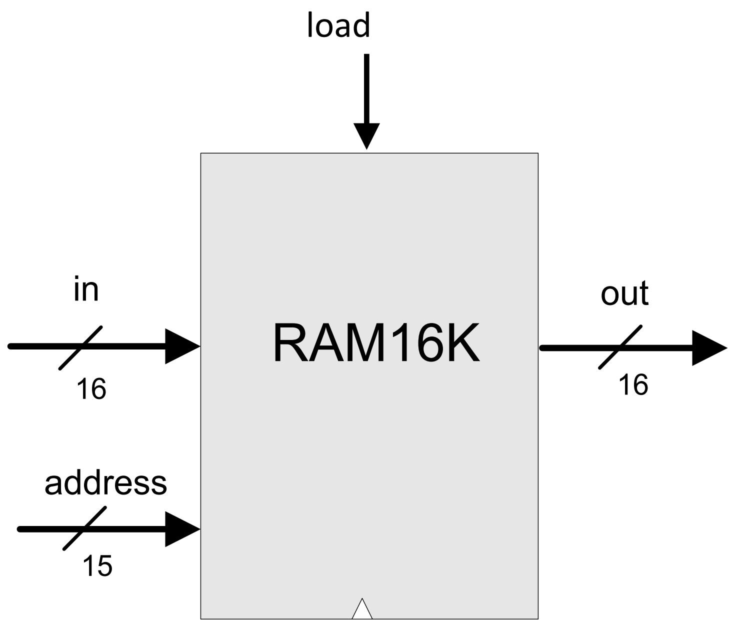
More about `peek` and `poke` this later in the course, when we'll write the OS.

Data memory

<u>Low-level (hardware) read/write logic:</u>

To read `RAM[k]`: set `address` to k,
probe `out`

To write `RAM[k]=x`: set `address` to k,
set `in` to x,
set `load` to 1,
run the clock



<u>High-level (OS) read/write logic:</u>

To read `RAM[k]`: use the OS command `out = peek(k)`

To write `RAM[k]=x`: use the OS command `poke(k,x)`

`peek` and `poke` are OS commands whose implementation should effect the same behavior as the low-level commands

More about `peek` and `poke` this later in the course, when we'll write the OS.

Lecture / construction plan

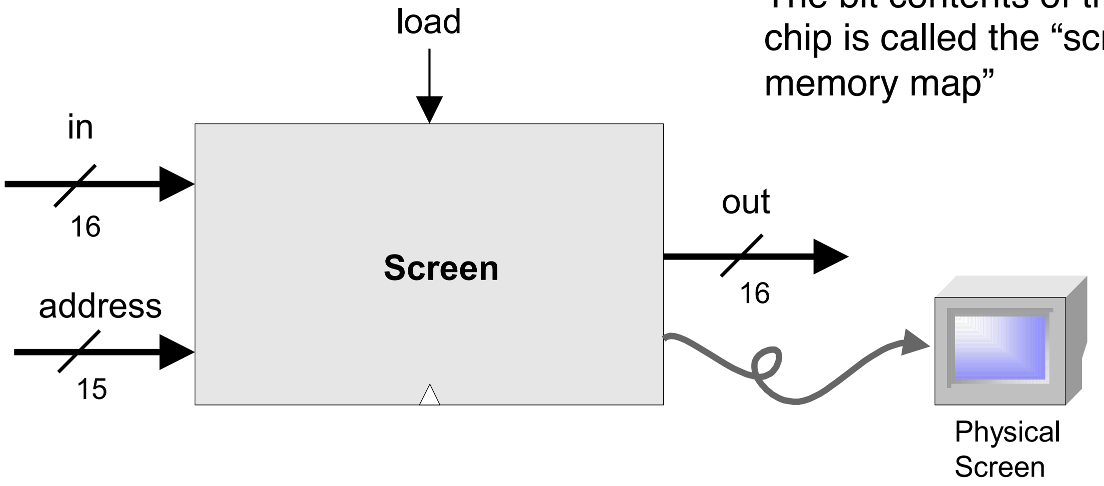
- Instruction memory
- Memory:
  - Data memory
  - Screen
  - Keyboard
- CPU
- Computer

Screen output

- How can we use the contents of memory to produce something on the screen?
- What do you think it looks like?
  - What do we need to do to make it work?
  - What sort of real world activity do we need to support?
- Small group work!

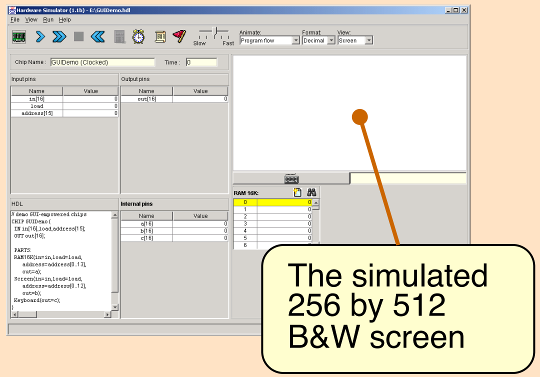
Worksheet
Question 2

Screen



The bit contents of the Screen chip is called the “screen memory map”

Simulated screen:



When loaded into the hardware simulator, the built-in Screen.hdl chip opens up a screen window; the simulator then refreshes this window from the screen memory map several times each second.

The Screen chip has a basic RAM chip functionality:

- read logic: `out = Screen[address]`
- write logic: `if load then Screen[address] = in`

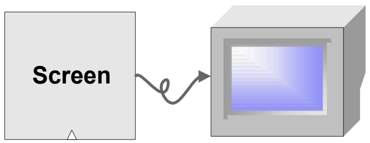
<u>Side effect:</u>

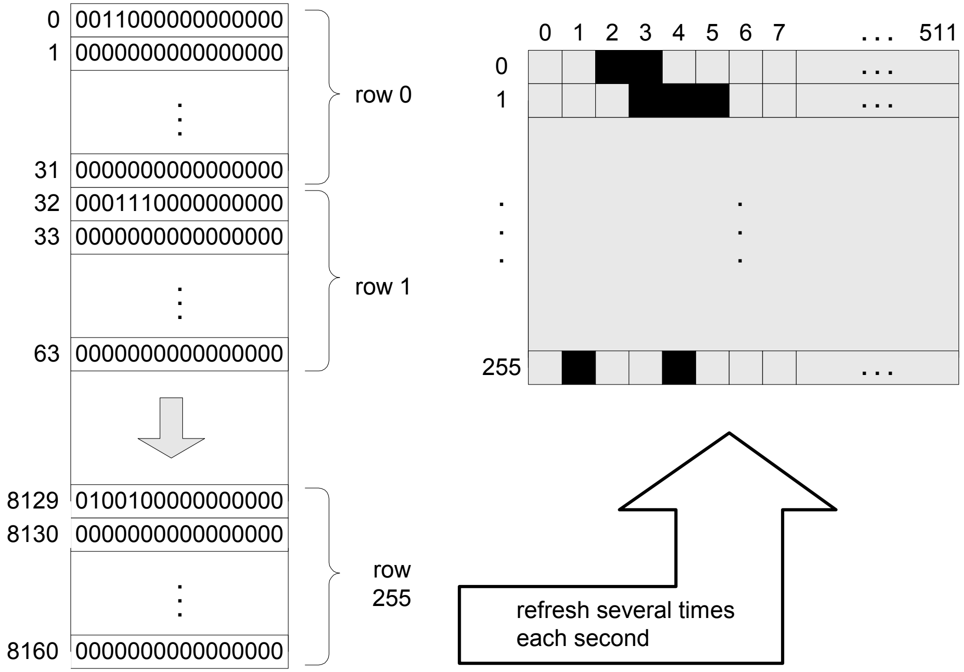
Continuously refreshes a 256 by 512 black-and-white screen device

Worksheet question 3

Screen memory map

In the Hack platform, the screen is implemented as an 8K 16-bit RAM chip.





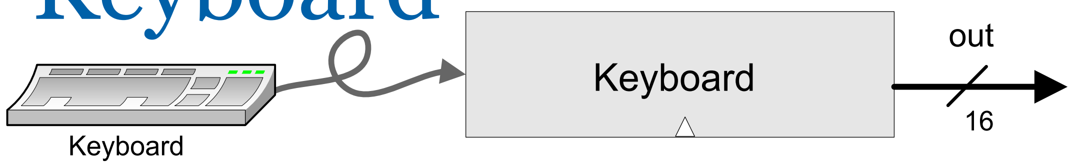
<u>How to set the `(row,col)` pixel of the screen to black or to white:</u>

- Low-level (machine language): Set the `col%16` bit of the word found at `Screen[row*32+col/16]` to `1` or to `0` (`col/16` is integer division)
- High-level: Use the OS command `drawPixel(row,col)` (effects the same operation, discussed later in the course, when we'll write the OS).

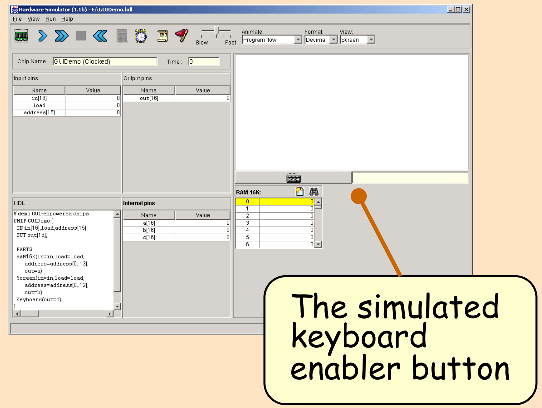
Keyboard

- How are we going to get input from the keyboard?
  - How do we use it?
  - What sort of activity do we need to support?

Keyboard



Simulated keyboard:



The keyboard is implemented as a built-in **Keyboard.hdl** chip. When this java chip is loaded into the simulator, it connects to the regular keyboard and pipes the scan-code of the currently pressed key to the keyboard memory map.

Keyboard chip: a single 16-bit register

Input: scan-code (16-bit value) of the currently pressed key, or 0 if no key is pressed

Output: same

Special keys:

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

Worksheet question 4

How to read the keyboard:

- Low-level (hardware): probe the contents of the Keyboard chip
- High-level: use the OS command keyPressed() (effects the same operation)

Next week

- There is a lecture on Monday!
- There is a tutorial next week.
- You should read "Chapter 5" from the forums and continue working on assignment 2.
- Any questions? Ask on the forum or right now!

Next lecture

- You should read "Chapter 6" from the forums and keep working on at Assignment 2.
- Any questions? Ask on the forum or right now!