CSCE 312: Computer Organization

David Kebo Hougninou

Processor Architecture
Outline

1. Computer Architecture vs Computer Organization
2. Von Neumann vs Harvard architecture
3. Fetch-Execute Cycle
4. HACK Computer

The Big Picture
Architecture vs Organization

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Customer visible features for an artifact</td>
<td>• Includes underlying details as well</td>
</tr>
</tbody>
</table>

Key Points

• Computer Architecture exposes features of a computer that the programmer needs to know in order to use it.

• The underlying details (programmer visible features or otherwise) constitute a computer’s organization.

• They are important to the designer of the computer, but not all are needed to be made visible to the programmer.

In this chapter we will discuss the hack architecture with its organization based on:

1. Building block components (Chapters 1-3)
2. Instruction set definition (Chapter 4)
Harvard Architecture (used by the Hack computer):
02 separate memory units: Instruction memory + Data memory

**Advantages:**
Separate instruction and data memory
We can address and manipulate each simultaneously

**Disadvantages:**
Two memory chips instead of one in von Neumann architecture
The size of the two chips is fixed vs flexible in von Neumann architecture

In stored program computer, computer operation at a high level amounts to the following two steps

1. Fetch the instruction from the instruction memory
2. Execute the instruction
Hack Computer

Abstraction:
A computer capable of running programs written in the Hack machine language

Implementation:
Built from the Hack chip-set.

Hack CPU

CPU abstraction:
Executes a Hack instruction and figures out which instruction to execute next

CPU Implementation:
To be discussed in the next module
Hack Computer

Memory
Memory: abstraction

- Address 0 to 16383: data memory
- Address 16384 to 24575: screen memory map
- Address 24576: keyboard memory map

The Hack RAM is realized by the RAM16K chip implemented in project 3.
Screen

To set pixel (row, col) on/off:

1. word = RAM[16384 + 32*row + col/16]
2. Set the (col % 16)th bit of word to 0 or 1
3. RAM[i] = word
Screen

- The Hack screen is realized by a built-in chip named Screen
- Screen: a regular RAM + display output side-effect.

Keyboard
Keyboard memory map

The Keyboard chip emits the scan-code of the currently pressed key, or 0 if no key is pressed.

The Hack character set

<table>
<thead>
<tr>
<th>key</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td>32</td>
</tr>
<tr>
<td>}</td>
<td>123</td>
</tr>
<tr>
<td>(</td>
<td>40</td>
</tr>
<tr>
<td>)</td>
<td>41</td>
</tr>
<tr>
<td>@</td>
<td>64</td>
</tr>
<tr>
<td>#</td>
<td>35</td>
</tr>
<tr>
<td>$</td>
<td>36</td>
</tr>
<tr>
<td>%</td>
<td>37</td>
</tr>
<tr>
<td>&amp;</td>
<td>38</td>
</tr>
<tr>
<td>&lt;</td>
<td>59</td>
</tr>
<tr>
<td>=</td>
<td>61</td>
</tr>
<tr>
<td>&gt;</td>
<td>62</td>
</tr>
<tr>
<td>?</td>
<td>63</td>
</tr>
<tr>
<td>^</td>
<td>94</td>
</tr>
<tr>
<td>_</td>
<td>95</td>
</tr>
<tr>
<td>`</td>
<td>96</td>
</tr>
<tr>
<td>&lt;space&gt;</td>
<td>32</td>
</tr>
<tr>
<td>!</td>
<td>33</td>
</tr>
<tr>
<td>“</td>
<td>34</td>
</tr>
<tr>
<td>9</td>
<td>57</td>
</tr>
<tr>
<td>Z</td>
<td>90</td>
</tr>
</tbody>
</table>

key code

<table>
<thead>
<tr>
<th>key code</th>
</tr>
</thead>
<tbody>
<tr>
<td>\newline</td>
</tr>
<tr>
<td>\backspace</td>
</tr>
<tr>
<td>\left arrow</td>
</tr>
<tr>
<td>\up arrow</td>
</tr>
<tr>
<td>\right arrow</td>
</tr>
<tr>
<td>\down arrow</td>
</tr>
<tr>
<td>\home</td>
</tr>
<tr>
<td>\end</td>
</tr>
<tr>
<td>\Page up</td>
</tr>
<tr>
<td>\Page down</td>
</tr>
<tr>
<td>\insert</td>
</tr>
<tr>
<td>\delete</td>
</tr>
<tr>
<td>\esc</td>
</tr>
<tr>
<td>f1</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>f12</td>
</tr>
</tbody>
</table>

Credit: www.nand2tetris.org
Keyboard

- Realized by a built-in chip named Keyboard
- Keyboard: A read-only 16-bit register + a keyboard input side-effect.

Memory implementation

Implementation outline:
- Uses the three chip-parts RAM16K, Screen, and Keyboard (as just described)
- Routes the address input to the correct address input of the relevant chip-part.
Hack Computer

To run a program on the Hack computer:
1. Load the program into the Instruction Memory
2. Press “reset”
3. The program starts running.
Loading a program into the Instruction Memory:

**Hardware implementation:**

plug-and-play ROM chips  (each comes pre-loaded with a program's code)

**Hardware simulation:** programs are stored in text files; the simulator's software features a load-program service.

The Hack Instruction Memory is realized by a built-in chip named ROM32K

ROM32K: a read-only, 16-bit, 32K RAM chip + program loading side-effect.
Hack Computer implementation

Diagram:
- ROM 32K
- CPU
- RAM
- Instruction flow:
  - Instruction from ROM to CPU
  - Reset to CPU
- Data flow:
  - Write to CPU
  - Output from CPU
  - Address from CPU

Diagram:
- ROM 32K
- CPU
- RAM
- Instruction flow:
  - Instruction from ROM to CPU
  - Reset to CPU
- Data flow:
  - Write to CPU
  - Output from CPU
  - Address from CPU
Hack Computer implementation

ROM

32K

CPU

RAM

 reset

 instruction

 pc

 inM

 writeM

 outM

 addressM

 Hack Computer implementation

ROM

32K

CPU

RAM

 reset

 instruction

 pc

 inM

 writeM

 outM

 addressM

 Hack Computer implementation

ROM

32K

CPU

RAM

 reset

 instruction

 pc

 inM

 writeM

 outM

 addressM