Learn 8-bit machine language with the **Toy CPU** emulator

An emulator in the style of the Altair 8800 or IMSAI 8080

Jim Hall (FreeDOS)
About me

Jim Hall
The FreeDOS Project
jhall@freedos.org

https://github.com/freedosproject/toycpu
MIS 100
Fundamentals of Information Technology in Organizations

4 undergraduate credits
Effective August 24, 2002 to present
Meets graduation requirements for
- General Education and Liberal Studies Elective

This course is the first information technology foundation course in the College of Management. It focuses on the technology literacy, managerial and business problem solving dimensions of computer based information systems. It provides students with an introduction to the fundamental terminology of the hardware, software and the people involved with computer based information systems. The course includes hands on computer lab time to introduce students to word processing, database, spreadsheet, and Internet microcomputer applications. This course is designed specifically to prepare students for information technology competence as needed in College of Management courses.

https://www.metrostate.edu/academics/courses/mis-100
• Students will be able to show competence in understanding Operating Systems and Utility Programs.
• Students will be able to show competence in understanding storage, both primary and secondary.
• Students will be able to show competence in understanding Computer Security and Safety, Ethics, and Privacy.
• Students will be able to show competence in understanding Database Management concepts.
• Students will be able to show competence in understanding basic concepts in Enterprise Computing.
• Students will be able to show competence in understanding Information System Development and Programming Languages.
• Students will also be able to use conditional formatting, import data from an external site and
Toy Machine Simulator

Program:

LOAD 1
ADD 2
STORE sum
PRINT sum
STOP
sum 0

Output:

3

stopped

Syntax reminder

get get a number from keyboard into accumulator
print print contents of accumulator
load Val load accumulator with Val (Val unchanged)
store M store contents of accumulator into memory location called M (accumulator unchanged)
add Val add Val to contents of accumulator (Val unchanged)
sub Val subtract Val from contents of accumulator (Val unchanged)
goto L go to instruction labeled L
ifpos L go to instruction labeled L if accumulator is >= zero
ifzero L go to instruction labeled L if accumulator is zero
stop stop running
M Num before program runs, set this memory location (called M) to numeric value Num

http://kernighan.com/toysim.html
Toy CPU Simulator

Counter

Instruction

Accumulator

Input mode: $ - counter | Enter - edit | R - run | Q - quit
Edit mode: + - bit | Space - flip | Enter - done

https://github.com/freedosproject/toycpu
## Toy CPU instruction set

<table>
<thead>
<tr>
<th>Binary Code</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>STOP</td>
</tr>
<tr>
<td>00000001</td>
<td>RIGHT</td>
</tr>
<tr>
<td>00000010</td>
<td>LEFT</td>
</tr>
<tr>
<td>00001111</td>
<td>NOT</td>
</tr>
<tr>
<td>00010001</td>
<td>AND</td>
</tr>
<tr>
<td>00010010</td>
<td>OR</td>
</tr>
<tr>
<td>00010011</td>
<td>XOR</td>
</tr>
<tr>
<td>00010100</td>
<td>LOAD</td>
</tr>
<tr>
<td>00010101</td>
<td>STORE</td>
</tr>
<tr>
<td>00010110</td>
<td>ADD</td>
</tr>
<tr>
<td>00010111</td>
<td>SUB</td>
</tr>
<tr>
<td>00011000</td>
<td>GOTO</td>
</tr>
<tr>
<td>00011001</td>
<td>IFZERO</td>
</tr>
<tr>
<td>10000000</td>
<td>NOP</td>
</tr>
</tbody>
</table>
Blink right, left, all lights

0. LOAD
1. “right”
2. LOAD
3. “left”
4. LOAD
5. “all”
6. STOP
7. ●●●●●●●● “right”
8. ●●●●●●●● “left”
9. ●●●●●●●●● “all”
Blink right, left, all lights

0. LOAD
1. “right”
2. LOAD
3. “left”
4. LOAD
5. “all”
6. STOP
7. 00000000 “right”
8. 00000000 “left”
9. 00000000 “all”

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000010100</td>
<td>LOAD</td>
</tr>
<tr>
<td>1</td>
<td>000000111</td>
<td>mem 7</td>
</tr>
<tr>
<td>2</td>
<td>000101000</td>
<td>LOAD</td>
</tr>
<tr>
<td>3</td>
<td>000010000</td>
<td>mem 8</td>
</tr>
<tr>
<td>4</td>
<td>000101000</td>
<td>LOAD</td>
</tr>
<tr>
<td>5</td>
<td>000010001</td>
<td>mem 9</td>
</tr>
<tr>
<td>6</td>
<td>000000000</td>
<td>STOP</td>
</tr>
<tr>
<td>7</td>
<td>000011111</td>
<td>“right”</td>
</tr>
<tr>
<td>8</td>
<td>111100000</td>
<td>“left”</td>
</tr>
<tr>
<td>9</td>
<td>111111111</td>
<td>“all”</td>
</tr>
</tbody>
</table>
Blink right, left, all lights

0. LOAD
1. “right”
2. NOT
3. NOP (so I don’t have to move it)
4. OR
5. “right”
6. STOP
7. ●●●●○○○○ “right”
8. ●●●●●●●● “left”
9. ●●●●●●●● “all”
Blink right, left, all lights

0. LOAD
1. “right”
2. NOP (so I don’t have to move 7)
3. NOT
4. OR
5. “right”
6. STOP
7. ●●●●●●●● “right”
8. ●●●●●●●● “left”
9. ●●●●●●●● “all”

<table>
<thead>
<tr>
<th>0</th>
<th>00010100</th>
<th>LOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00000111</td>
<td>mem 7</td>
</tr>
<tr>
<td>2</td>
<td>10000000</td>
<td>NOP</td>
</tr>
<tr>
<td>3</td>
<td>00001111</td>
<td>NOT</td>
</tr>
<tr>
<td>4</td>
<td>00010010</td>
<td>OR</td>
</tr>
<tr>
<td>5</td>
<td>00000111</td>
<td>mem 7</td>
</tr>
<tr>
<td>6</td>
<td>00000000</td>
<td>STOP</td>
</tr>
<tr>
<td>7</td>
<td>00001111</td>
<td>“right”</td>
</tr>
<tr>
<td>8</td>
<td>11110000</td>
<td>“left”</td>
</tr>
<tr>
<td>9</td>
<td>11111111</td>
<td>“all”</td>
</tr>
</tbody>
</table>
Countdown

0. LOAD
1. “start value”
2. IFZERO
3. “end”
4. SUB
5. “one”
6. GOTO
7. “beginning”
8. STOP
9. “start value” (3)
10. “one” (1)
Countdown 3…2…1…0

0. LOAD
1. “start value”
2. IFZERO
3. “end”
4. SUB
5. “one”
6. GOTO
7. “beginning”
8. STOP
9. “start value” (3)
10. “one” (1)
Move a light to the right

0. LOAD
1. “start value”
2. IFZERO
3. “end”
4. RIGHT
5. GOTO
6. “loop start”
7. STOP
8. ●○○○○○○○ “start value”
Move a light to the right

0. LOAD
1. “start value”
2. IFZERO
3. “end”
4. RIGHT
5. GOTO
6. “loop start”
7. STOP
8. ●●●●●●●●●● “start value”

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00010100</td>
<td>LOAD</td>
</tr>
<tr>
<td>1</td>
<td>00001000</td>
<td>mem 8</td>
</tr>
<tr>
<td>2</td>
<td>00011001</td>
<td>IFZERO</td>
</tr>
<tr>
<td>3</td>
<td>00000111</td>
<td>“end”</td>
</tr>
<tr>
<td>4</td>
<td>00000001</td>
<td>RIGHT</td>
</tr>
<tr>
<td>5</td>
<td>00011000</td>
<td>GOTO</td>
</tr>
<tr>
<td>6</td>
<td>00000010</td>
<td>mem 2</td>
</tr>
<tr>
<td>7</td>
<td>00000000</td>
<td>STOP</td>
</tr>
<tr>
<td>8</td>
<td>10000000</td>
<td>“start value”</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Add 1 + 2

0. LOAD
1. “first”
2. ADD
3. “second”
4. STORE
5. “sum”
6. STOP
7. •••••••• • “first (1)”
8. •••••••• • “second (2)”
9. ••••••••• • “sum”
Add 1 + 2

<table>
<thead>
<tr>
<th></th>
<th>LOAD</th>
<th>“first”</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADD</td>
<td>“second”</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>STORE</td>
<td>“sum”</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>STOP</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>“first (1)”</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>“second (2)”</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>“sum”</td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>00010100</th>
<th>LOAD</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00000111</td>
<td>mem 7</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>00010110</td>
<td>ADD</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>00001000</td>
<td>mem 8</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>00010101</td>
<td>STORE</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>00001001</td>
<td>mem 9</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>00000000</td>
<td>STOP</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>00000001</td>
<td>“first (1)”</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>00000010</td>
<td>“second (2)”</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>11111111</td>
<td>“sum”</td>
<td>9</td>
</tr>
</tbody>
</table>
```c
#include <stdio.h>

int main() {
    int sum;
    sum = 1 + 2;
    printf("1 + 2 = %d\n", sum);
    return 0;
}
```
About me

Jim Hall
The FreeDOS Project
jhall@freedos.org

https://github.com/freedosproject/toycpu