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)

FOSDEM'23
Brussels / 4 & 5 February 2023

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, spread sheet, 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

https://www.metrostate.edu/academics/courses/mis-100

Toy Machine Simulator

Program:

```
   LOAD  1
   ADD   2
   STORE sum
   PRINT sum
   STOP
sum      0
```

Output:

```
3

stopped
```

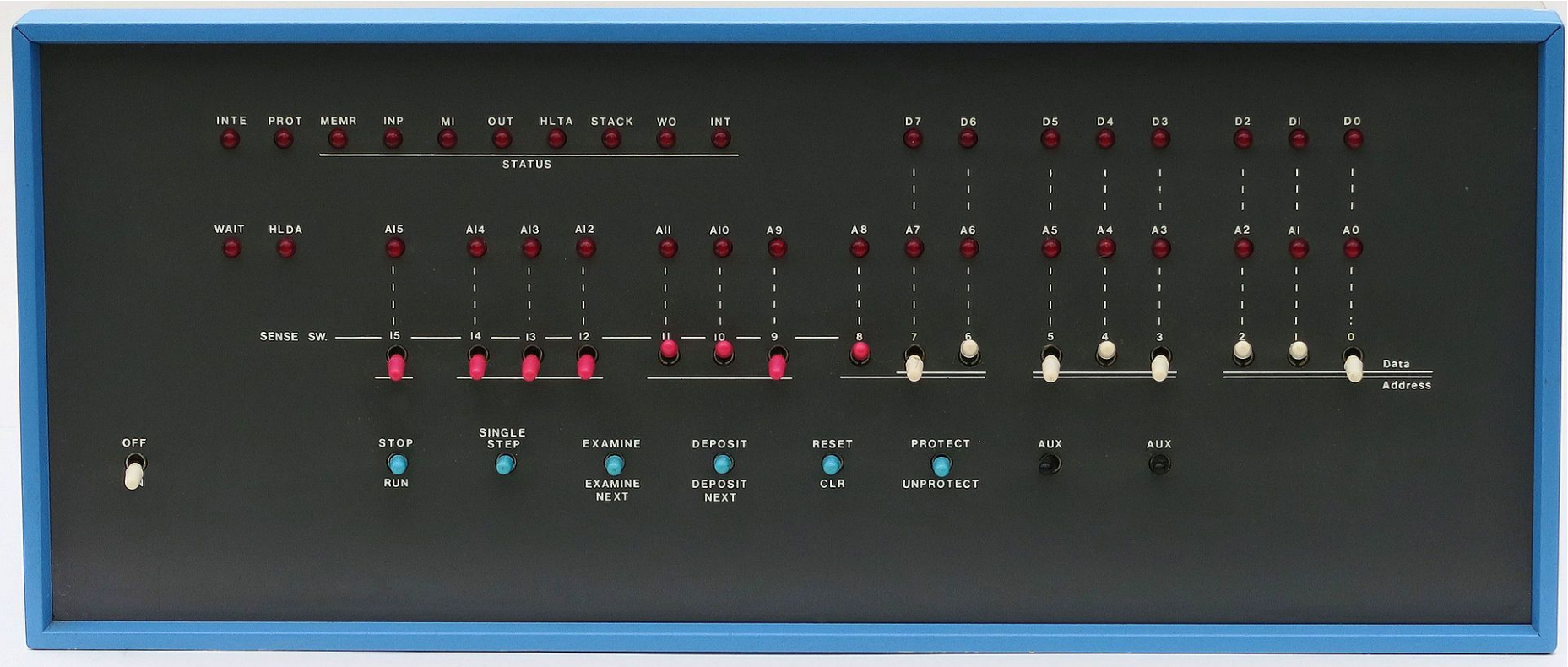
Accumulator: 3

RUN

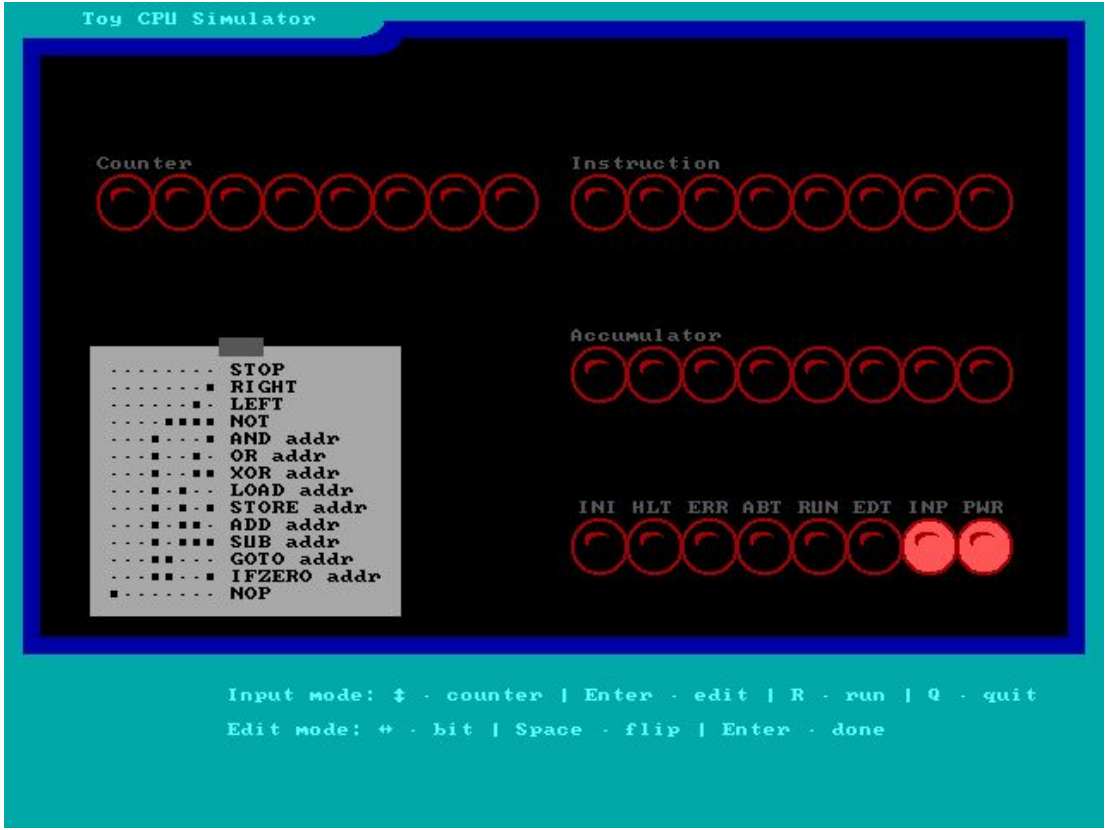
Syntax reminder

```
   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



Wikipedia | photo of Altair 8800 by Cromemco (CC BY-SA)



https://github.com/freedosproject/toycpu

Toy CPU instruction set

00000000	STOP	00010100	LOAD
00000001	RIGHT	00010101	STORE
00000010	LEFT	00010110	ADD
00001111	NOT	00010111	SUB
00010001	AND	00011000	GOTO
00010010	OR	00011001	IFZERO
00010011	XOR	10000000	NOP

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. ○○○○●●●● “right”
8. ●●●●○○○○ “left”
9. ●●●●●●●● “all”

0	00010100	LOAD
1	00000111	mem 7
2	00010100	LOAD
3	00001000	mem 8
4	00010100	LOAD
5	00001001	mem 9
6	00000000	STOP
7	00001111	“right”
8	11110000	“left”
9	11111111	“all”

Blink right, left, all lights

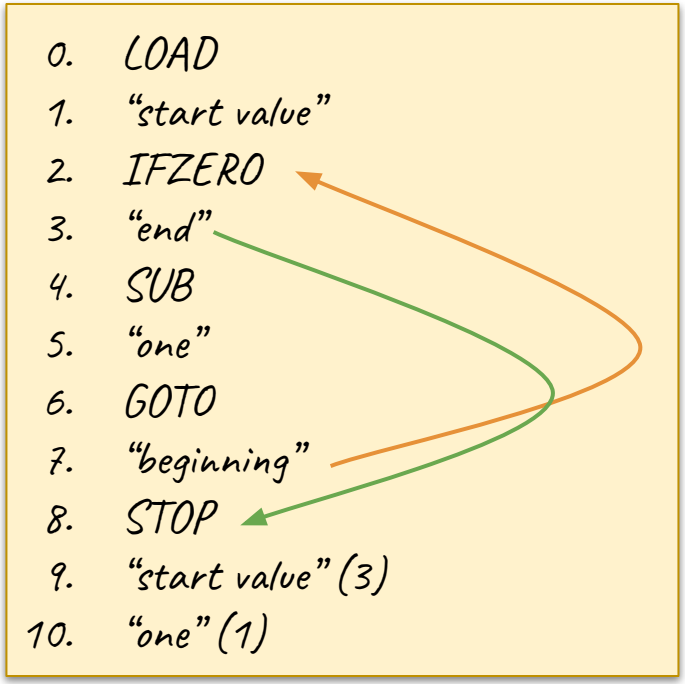
0. LOAD
1. “right”
2. NOT
3. NOP (so I don’t have to move 7)
4. OR
5. “right”
6. STOP
7. ○○○○●●●● “right”
8. ~~●●●●○○○○ “left”~~
9. ~~●●●●●●●● “all”~~

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”~~

0	00010100	LOAD
1	00000111	mem 7
2	10000000	NOP
3	00001111	NOT
4	00010010	OR
5	00000111	mem 7
6	00000000	STOP
7	00001111	“right”
~~8~~	~~11110000~~	~~“left”~~
~~9~~	~~11111111~~	~~“all”~~

Countdown

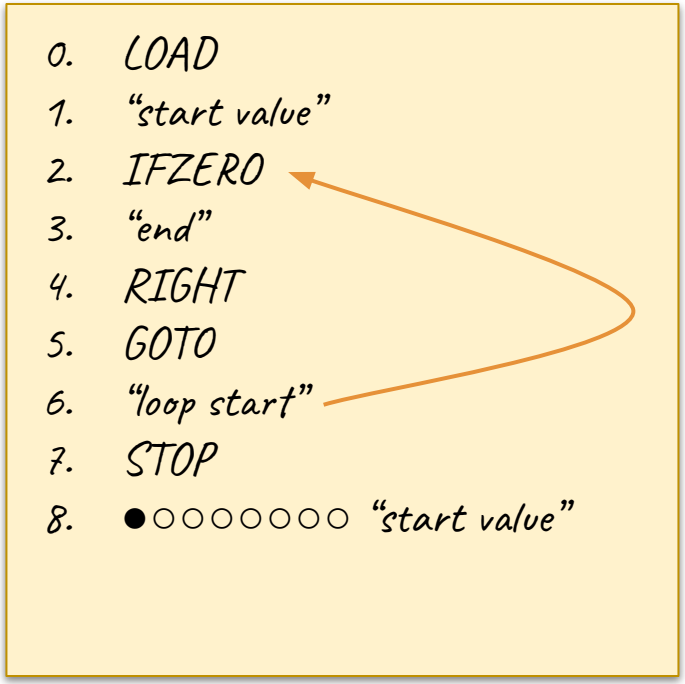

0. LOAD
1. "start value"
2. IFZERO
3. "end"
4. SUB
5. "one"
6. GOTO
7. "beginning"
8. STOP
9. "start value" (3)
10. "one" (1)

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)

0	00010100	LOAD
1	00001001	mem 9
2	00011001	IFZERO
3	00001000	mem 8
4	00010111	SUB
5	00001010	mem 10
6	00011000	GOTO
7	00000010	mem 2
8	00000000	STOP
9	00000011	“start value” (3)
10	00000001	“one” (1)

Move a light to the right



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”

0	00010100	LOAD
1	00001000	mem 8
2	00011001	IFZERO
3	00000111	“end”
4	00000001	RIGHT
5	00011000	GOTO
6	00000010	mem 2
7	00000000	STOP
8	10000000	“start value”
9		

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

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

0	00010100	LOAD
1	00000111	mem 7
2	00010110	ADD
3	00001000	mem 8
4	00010101	STORE
5	00001001	mem 9
6	00000000	STOP
7	00000001	“first (1)”
8	00000010	“second (2)”
9	11111111	“sum”

Add 1 + 2

C

```c
#include <stdio.h>

int main()
{
  int sum;

  sum = 1 + 2;
  printf("1 + 2 = %d\n", sum);

  return 0;
}
```

F77

```
      PROGRAM ADD
      INTEGER SUM
      SUM = 1 + 2
      PRINT *, '1 + 2 =', SUM
      END
```

About me

Jim Hall

The FreeDOS Project

jhall@freedos.org

https://github.com/freedosproject/toycpu