5. The TOY Machine

What is TOY?

An imaginary machine similar to:
- Ancient computers.
- Today’s microprocessors.

Why Study TOY?

Machine language programming.
- How do Java programs relate to computer?
- Key to understanding Java references.
- Still situations today where it is really necessary.
  - multimedia, computer games, scientific computing, SSE, AVX

Computer architecture.
- How does it work?
- How is a computer put together?

TOY machine. Optimized for simplicity, not cost or performance.

Inside the Box

Switches. Input data and programs.
- Fastest form of storage.
- Scratch space during computation.
- 16 16-bit registers.
- Register 0 is always 0.

Lights. View data.

Memory.
- Stores data and programs.
- 256 16-bit "words."
- Special word for stdin / stdout.

Arithmetic-logic unit (ALU). Manipulate data stored in registers.

Program counter (PC).
- An extra 8-bit register.
- Keeps track of next instruction to be executed.

Standard input, standard output. Interact with outside world.
Data and Programs Are Encoded in Binary

Each bit consists of two states:
- 1 or 0; true or false.
- Switch is on or off; wire has high voltage or low voltage.

Everything stored in a computer is a sequence of bits.
- Data and programs.
- Text, documents, pictures, sounds, movies, executables, ...

\[ M = 77_{10} = 01001101_{2} = 4D_{16} \]
\[ O = 79_{10} = 01001111_{2} = 4F_{16} \]

Binary Encoding

How to represent integers?
- Use binary encoding.
- Ex: \( 6375_{10} = 0001100011100111_{2} \)

<table>
<thead>
<tr>
<th>Dec</th>
<th>Bin</th>
<th>Dec</th>
<th>Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>15</td>
<td>1111</td>
</tr>
</tbody>
</table>

Hexadecimal Encoding

How to represent integers?
- Use hexadecimal encoding.
- Binary code, four bits at a time.
- Ex: \( 6375_{10} = 0001100011100111_{2} \) \( \equiv 18E7_{16} \)

<table>
<thead>
<tr>
<th>Dec</th>
<th>Bin</th>
<th>Hex</th>
<th>Dec</th>
<th>Bin</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0</td>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>1</td>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>2</td>
<td>10</td>
<td>1010</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>3</td>
<td>11</td>
<td>1011</td>
<td>B</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>4</td>
<td>12</td>
<td>1100</td>
<td>C</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>5</td>
<td>13</td>
<td>1101</td>
<td>D</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>6</td>
<td>14</td>
<td>1110</td>
<td>E</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>7</td>
<td>15</td>
<td>1111</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dec</th>
<th>Bin</th>
<th>Dec</th>
<th>Bin</th>
<th>Dec</th>
<th>Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ 6375_{10} = 2^{12} + 2^{11} + 2^{7} + 2^{6} + 2^{5} + 2^{2} + 2^{1} + 2^{0} \]
\[ = 4096 + 2048 + 128 + 64 + 32 + 4 + 2 + 1 \]
Machine "Core" Dump

Machine contents at a particular place and time.
- Record of what program has done.
- Completely determines what machine will do.

A Sample Program

Program counter. The pc is initially 10, so the machine interprets 8A00 as an instruction.

Load. [opcode 8]
- Loads the contents of some memory location into a register.
- 8A00 means load the contents of memory cell 00 into register A.
Load. [opcode 8]
- Loads the contents of some memory location into a register.
- 8B01 means load the contents of memory cell 01 into register B.

Add. [opcode 1]
- Add contents of two registers and store sum in a third.
- 1CAB means add the contents of registers A and B and put the result into register C.

Store. [opcode 9]
- Stores the contents of some register into a memory cell.
- 9C02 means store the contents of register C into memory cell 02.

Halt. [opcode 0]
- Stop the machine.
Program and Data

Program. Sequence of 16-bit integers, interpreted one way.

Data. Sequence of 16-bit integers, interpreted other way.

Program counter (pc). Holds memory address of the "next instruction" and determines which integers get interpreted as instructions.

16 instruction types. Changes contents of registers, memory, and pc in specified, well-defined ways.

TOY Instruction Set Architecture

TOY instruction set architecture (ISA).
- Interface that specifies behavior of machine.
- 16 register, 256 words of main memory, 16-bit words.
- 16 instructions.

Each instruction consists of 16 bits.
- Bits 12-15 encode one of 16 instruction types or opcodes.
- Bits 8-11 encode destination register d.
- Bits 0-7 encode:

  - [Format 1] source registers s and t
  - [Format 2] 8-bit memory address or constant

Interfacing with the TOY Machine

To enter a program or data:
- Set 8 memory address switches.
- Set 16 data switches.
- Press Load: data written into addressed word of memory.

To view the results of a program:
- Set 8 memory address switches.
- Press Look: contents of addressed word appears in lights.

Using the TOY Machine: Run

To run the program:
- Set 8 memory address switches to address of first instruction.
- Press Look to set pc to first instruction.
- Press Run button to repeat fetch-execute cycle until halt opcode.

Fetch-execute cycle.
- Fetch: get instruction from memory.
- Execute: update pc, move data to or from memory and registers, perform calculations.
Flow Control

Flow control.
- To harness the power of TOY, need loops and conditionals.
- Manipulate pc to control program flow.

Branch if zero. [opcode C]
- Changes pc depending on whether value of some register is zero.
- Used to implement: for, while, if-else.

Branch if positive. [opcode D]
- Changes pc depending on whether value of some register is positive.
- Used to implement: for, while, if-else.

Multiply.

Given integers a and b, compute c = a × b.

TOY multiplication. No direct support in TOY hardware.

Brute-force multiplication algorithm:
- Initialize c to 0.
- Add b to c, a times.

Issues ignored. Slow, overflow, negative numbers.

Brute force multiply in Java

```java
int a = 3;
int b = 9;
int c = 0;
while (a != 0) {
    c = c + b;
    a = a - 1;
}
```

Step-By-Step Trace

```
0A: 0003   3 ← inputs
0B: 0009   9
0C: 0000   0 ← output
0D: 0000   0 ← constants
0E: 0001   1
10: 8A0A   RA ← mem[OA] 0003
11: 8B0B   RB ← mem[OB] 0009
12: 8C0D   RC ← mem[OD] 0000
13: 810E   R1 ← mem[OE] always 1
14: CA18   if (RA == 0) pc ← 18 a
15: 1CCB   RC ← RC + RB c = c + b
16: 2AA1   RA ← RA - R1 a = a - 1
17: C014   pc ← 14
18: 9C0C   mem[OC] ← RC
19: 0000   halt
```

```
0A: 0003
0B: 0009
0C: 0000
0D: 0000
0E: 0001
10: 8A0A   RA ← mem[OA]
11: 8B0B   RB ← mem[OB]
12: 8C0D   RC ← mem[OD]
13: 810E   R1 ← mem[OE]
14: CA18   if (RA == 0) pc ← 18
15: 1CCB   RC ← RC + RB
16: 2AA1   RA ← RA - R1
17: C014   pc ← 14
18: 9C0C   mem[OC] ← RC
19: 0000   halt
```

```
R1  RA  RB  RC
0003  0009  0000
0001
0002
0012
0001
0000
001B
```

```
int a = 3;
int b = 9;
int c = 0;
while (a != 0) {
    c = c + b;
    a = a - 1;
}
halt
```

```
multiply.toy
```

```
multiply.toy
```

A Little History

Electronic Numerical Integrator and Calculator (ENIAC).
- First widely known general purpose electronic computer.
- Conditional jumps, programmable.
- Programming: change switches and cable connections.
- Data: enter numbers using punch cards.

30 tons
30 x 50 x 8.5 ft
17,468 vacuum tubes
300 multiply/sec

TOY is a general-purpose computer.
- Sufficient power to perform any computation.
- Limited only by amount of memory and time.

Stored-program computer. [von Neumann memo, 1944]
- Data and program encoded in binary.
- Data and program stored in same memory.
- Can change program without rewiring.

Outgrowth of Alan Turing's work. (stay tuned)

Harvard vs. Princeton

Harvard architecture.
- Separate program and data memories.
- Can’t load game from disk (data) and execute (program).
- Used in some microcontrollers.

Von Neumann architecture.
- Program and data stored in same memory.
- Used in almost all computers.

Q. What’s the difference between Harvard and Princeton?
A. At Princeton, data and programs are the same.