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 a computer?
  - Key to understanding Java references.
  - Still some situations today where it is really necessary.
- Computer architecture.
  - How does a computer put together?

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

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, ...

Binary Encoding

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

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

6375\(_{10} = 2^15 + 2^{13} + 2^8 + 2^5 + 2^3 + 2^2 + 2^1 + 2^0\)

= 4096 + 2048 + 128 + 64 + 32 + 8 + 2 + 1

Hexadecimal Encoding

How to represent integers?
- Use hexadecimal encoding.
- Ex: 6375\(_{10} = 0001100011100111\) \(_{16}\)

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

6375\(_{10} = 1 \times 16^3 + 8 \times 16^2 + 14 \times 16^1 + 7 \times 16^0\)

= 4096 + 2048 + 224 + 7
Inside the TOY Box

Switches: Input data and programs.
Lights: View data.
Memory:
- Stores data and programs.
- 256 16-bit "words."
- Special word for stdin / stdout.
Program counter (PC):
- An extra 8-bit register.
- Next instruction to be executed.

Registers
- Fastest form of storage.
- Register 0 is always 0.

A Sample Program

A sample program. Adds 0008160005160000.

Load

Loads the contents of some memory location into a register.
8A00 means load the contents of memory cell 00 into register A.

A core dump is the contents of machine 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.

Binary People

There are only 10 types of people in the world: Those who understand binary and those who don’t.

http://www.thinkgeek.com/tshirts

15/10/2012
Load. (opcode 8)

- Loads the contents of some memory location into a register.
- EA0 means load the contents of memory cell a0 into register a.

<table>
<thead>
<tr>
<th>RA</th>
<th>RB</th>
<th>RC</th>
<th>pc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0008</td>
<td>0000</td>
<td>0000</td>
<td>11</td>
</tr>
</tbody>
</table>

Registers | Program counter
---|---

Add. (opcode 1)

- Add contents of two registers and store sum in a third.
- I1c0a means add the contents of registers a and b and put the result into register c.

<table>
<thead>
<tr>
<th>RA</th>
<th>RB</th>
<th>RC</th>
<th>pc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0008</td>
<td>0005</td>
<td>0000</td>
<td>12</td>
</tr>
</tbody>
</table>

Registers | Program counter
---|---

Store. (opcode 9)

- Stores the contents of some register into a memory cell.
- E0c02 means store the contents of register c into memory cell 0.

<table>
<thead>
<tr>
<th>RA</th>
<th>RB</th>
<th>RC</th>
<th>pc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0008</td>
<td>0005</td>
<td>0000</td>
<td>13</td>
</tr>
</tbody>
</table>

Registers | Program counter
---|---

Halt. (opcode 0)

- Stop the machine.

<table>
<thead>
<tr>
<th>RA</th>
<th>RB</th>
<th>RC</th>
<th>pc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0008</td>
<td>0005</td>
<td>0000</td>
<td>14</td>
</tr>
</tbody>
</table>

Registers | Program counter
---|---

**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.

<table>
<thead>
<tr>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>F</td>
</tr>
</tbody>
</table>

**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 a and b
  - [Format 2]: 8-bit memory address or constant

<table>
<thead>
<tr>
<th>Format 1</th>
<th>Format 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>dest d</td>
</tr>
<tr>
<td>opcode</td>
<td>dest a</td>
</tr>
</tbody>
</table>

15/10/2012
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.

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.

An Example: Multiplication

Multiply. Given integers a and b, compute \( c = a \times b \).

TOY multiplication. No direct support in TOY hardware.

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

Issues ignored. Slow, overflow, negative numbers.

Multiply

```
10: LADA RA ← mem[OA]  a = 3;
11: RDBR RB ← mem[OB]  b = 9;
12: RCD0 RC ← mem[OD]  c = 0;
13: R10E RI ← mem[OE]  always 1
14: CALL if (RA == 0) pc ← 18
15: ICBR RC ← RC + RB  c ← c + b;
16: ZAAL RA ← RA – RI   pc ← 14
17: C014 pc ← 14
18: WDCC mem[OC] ← RC
19: 0000 halt
```

Step-By-Step Trace

```
10: LADA RA ← mem[OA] 0003 0009 0009
11: RDBR RB ← mem[OB] 0000 0009 0009
12: RCD0 RC ← mem[OD] 0000 0009 0009
13: R10E RI ← mem[OE] 0001 0000 0001
14: CALL if (RA == 0) pc ← 18
15: ICBR RC ← RC + RB 0012 0012 0012
16: ZAAL RA ← RA – RI   0000 0000 0000
17: C014 pc ← 14
18: WDCC mem[OC] ← RC
19: 0000 halt
```
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.

20 ops
30 x 50 x 8.5 ft
17,468 vacuum tubes
300 multiply/sec
15,000 watts

Basic Characteristics of TOY Machine

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.

All modern computers are general-purpose computers and have same (von Neumann) architecture.

5: Supplemental Notes

Why do They Call it “Core”?

Register 0 always reads 0.

Loads from mem[FF] from stdin.

Stores to mem[FF] to stdout.

16-bit registers.

16-bit memory.

8-bit program counter.

TOY Reference Card

Format 1

<table>
<thead>
<tr>
<th>opcode</th>
<th>dest d</th>
<th>source s</th>
<th>source t</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: halt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: add</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: sub</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: and</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4: xor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5: shift left</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6: shift right</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7: load addr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8: load</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9: store</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A: load indirect</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B: store indirect</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C: branch zero</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D: branch positive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E: jump register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F: jump and link</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Register 0 always reads 0.

Loads from mem[FF] from stdin.

Stores to mem[FF] to stdout.

16-bit registers.

16-bit memory.

8-bit program counter.
An Efficient Multiplication Algorithm

**Inefficient multiply.**
- Brute force multiplication algorithm loops $2n$ times.
- In worst case, 65,535 additions!

**"Grade-school" multiplication.**
- Always 16 additions to multiply 16-bit integers.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>1101101010101111</td>
</tr>
<tr>
<td>2465</td>
<td>1011000000101000</td>
</tr>
<tr>
<td>3248</td>
<td>1101101101001111</td>
</tr>
<tr>
<td>17000</td>
<td>1010000011001111</td>
</tr>
</tbody>
</table>

Shift Left

**Shift left.** (opcode 5)
- Move bits to the left, padding with zeros as needed.
  - $1234_{16} \ll 7 = 1A00_{16}$

**Shift right.** (opcode 6)
- Move bits to the right, padding with sign bit as needed.
  - $1234_{16} \gg 7 = 0024_{16}$

Logical AND
- Logic operations are BITWISE.
  - $0024_{16} \& 0001_{16} = 0000_{16}$

Bitwise AND
- $x \& y = x \land y$
### Shifting and Masking

**Shift and mask:** get the 7th bit of 1234.
- Compute: 1234 $\gg$ 7 = 0034
- Compute: 0034 $\&$ 1 = 0

### Binary Multiplication

**OA:** 0001 3 inputs
**OB:** 0009 9 $\Phi$ output
**OC:** 0000 0 $\Phi$ output
**OD:** 0000 0 output
**OE:** 0001 1 constants

### Shift Right (Sign Extension)

**Shift right:** (opcode 6)
- Move bits to the right, padding with sign bit as needed.
- **FTDCR** $\gg$ 2 = FFFF
- **-5432** $\gg$ 2 = 13

### Bitwise XOR

**Bitwise XOR:** (opcode 4)
- Logic operations are BITWISE.
- 1234 $^\oplus$ FAD2 = E8E6

### 5: Extra Slides

**Jump absolute.**
- Jump to a fixed memory address.
- Branch if zero with destination.
- Register 0 is always 0

**Register assignment.**
- No instruction that transfers contents of one register into another.
- Pseudo-instruction that simulates assignment:
  - add with register 0 as one of two source registers

**No-op.**
- Instruction that does nothing.
- Plays the role of whitespace in C programs.
- Numerous other possibilities!
What is TOY?

An imaginary machine similar to:

- Ancient computers.