Stored Program Machines

Chris Gregg, Based on slides by Eric Roberts
CS 208E
October 16, 2017
The von Neumann Architecture

• One of the foundational ideas of modern computing—traditionally attributed to John von Neumann although others can make valid claims to the idea—is that code is stored in the same memory as data. This concept is called the *stored programming model*.

• The next few slides introduce the Manchester Baby, which was the first stored-program computer. In the rest of today’s class, we will describe the operation of a slightly more powerful machine that Eric Roberts nicknamed "Toddler". 

John von Neumann and J. Robert Oppenheimer
The Manchester Baby
Structure of the Toddler Machine

<table>
<thead>
<tr>
<th></th>
<th>0&lt;sub&gt;x&lt;/sub&gt;</th>
<th>1&lt;sub&gt;x&lt;/sub&gt;</th>
<th>2&lt;sub&gt;x&lt;/sub&gt;</th>
<th>3&lt;sub&gt;x&lt;/sub&gt;</th>
<th>4&lt;sub&gt;x&lt;/sub&gt;</th>
<th>5&lt;sub&gt;x&lt;/sub&gt;</th>
<th>6&lt;sub&gt;x&lt;/sub&gt;</th>
<th>7&lt;sub&gt;x&lt;/sub&gt;</th>
<th>8&lt;sub&gt;x&lt;/sub&gt;</th>
<th>9&lt;sub&gt;x&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>1</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>2</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>3</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>4</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>5</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>6</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>7</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>8</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
<tr>
<td>9</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
<td>+0000</td>
</tr>
</tbody>
</table>

AC
+0000

PC
00

IR
+0000

Console
# The Toddler Instruction Set

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx LOAD xx</td>
<td>Loads the value from address xx into the AC</td>
</tr>
<tr>
<td>2xx STORE xx</td>
<td>Stores the value from AC into address xx</td>
</tr>
<tr>
<td>3xx ADD xx</td>
<td>Adds the value at address xx to the AC</td>
</tr>
<tr>
<td>4xx SUB xx</td>
<td>Subtracts the value at address xx from AC</td>
</tr>
<tr>
<td>500 HALT</td>
<td>Halts the machine</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8xx INPUT xx</td>
<td>Reads a value into address xx</td>
</tr>
<tr>
<td>9xx OUTPUT xx</td>
<td>Prints the value in address xx</td>
</tr>
</tbody>
</table>
The Add-Two-Numbers Program

(01) +850 INPUT 50
(02) +851 INPUT 51
(03) +150 LOAD 50
(04) +351 ADD 51
(05) +252 STORE 52
(06) +952 OUTPUT 52
(07) +500 HALT
The Instruction Cycle

1. *Fetch the current instruction.* In this phase, Toddler finds the word from the memory address specified by the PC and copies its value into the IR.

2. *Increment the program counter.* Once the current instruction has been copied into the IR, Toddler adds one to the PC so that it points to the next instruction.

3. *Decode the instruction in the instruction register.* The value copied into the IR is a three-digit integer. To use it as an instruction, Toddler must divide the instruction word into its opcode and address components.

4. *Execute the instruction.* Once the operation code and address field have been identified, the Toddler processor must carry out the steps necessary to perform the indicated action.
The Add-Two-Numbers Program

<table>
<thead>
<tr>
<th></th>
<th>0x</th>
<th>1x</th>
<th>2x</th>
<th>3x</th>
<th>4x</th>
<th>5x</th>
<th>6x</th>
<th>7x</th>
<th>8x</th>
<th>9x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AC  +000
PC  00
IR  062

Console

? 17
? 25
42
# The Toddler Instruction Set

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx LOAD xx</td>
<td>Loads the value from address xx into the AC</td>
</tr>
<tr>
<td>2xx STORE xx</td>
<td>Stores the value from AC into address xx</td>
</tr>
<tr>
<td>3xx ADD xx</td>
<td>Adds the value at address xx to the AC</td>
</tr>
<tr>
<td>4xx SUB xx</td>
<td>Subtracts the value at address xx from AC</td>
</tr>
<tr>
<td>500 HALT</td>
<td>Halts the machine</td>
</tr>
<tr>
<td>5xx JUMP xx</td>
<td>Takes the next instruction from address xx</td>
</tr>
<tr>
<td>6xx JUMPZ xx</td>
<td>Jumps to xx if the AC is zero</td>
</tr>
<tr>
<td>7xx JUMPN xx</td>
<td>Jumps to xx if the AC is negative</td>
</tr>
<tr>
<td>8xx INPUT xx</td>
<td>Reads a value into address xx</td>
</tr>
<tr>
<td>9xx OUTPUT xx</td>
<td>Prints the value in address xx</td>
</tr>
</tbody>
</table>
The Countdown Program

**assembly language**

```
start:    LOAD ten
         STORE i

loop:     JUMPN done
         OUTPUT i
         LOAD i
         SUB one
         STORE i
         JUMP loop

done:     HALT

one:      1

ten:      10

i:        0
```

(01) +111
(02) +212
(03) +709
(04) +912
(05) +112
(06) +410
(07) +212
(08) +503
(09) +500
(10) +001
(11) +010
(12) +000
Representing Constants

• Just as was true for the Analytical Engine, constants in the Toddler machine need to be stored in one of the memory addresses, as illustrated by the following lines from the assembly language version of `Countdown.td`:

```
one:    1
ten:    10
```

• The instruction `LOAD ten` then refers to a memory address that contains the value 10.

• Toddler also allows you to write

```
LOAD #10
```

which finds space for the constant 10 at the end of the program and then fills in the `LOAD` instruction with the address of that constant.
Exercise: Multiply Two Numbers

- How would you write a Toddler program to multiply two nonnegative numbers, even though the machine has no multiply instruction?
The End