<table>
<thead>
<tr>
<th>Today</th>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>• LMC cont</td>
<td>• HW1 (out, due Tues)</td>
</tr>
</tbody>
</table>
Computer System Building Blocks

Basic Building Blocks of Computer Systems

- Hardware supports machine instructions
  - called the “Instruction Set Architecture” (ISA)
  - ARM, x86, x86-64 are examples

- Operating system provides abstractions over hardware (e.g., files, processes)
- And support for writing/running programs (e.g., libraries, execution)
- Programs in high-level languages converted to machine executable formats
Getting Started: Simple Model of a Computer

The LMC ("Little Man Computer")

- Simple “computer” model for learning architecture and machine-programming

The little man repeats the following steps

1. Read 2-digit number from instruction counter (e.g., 00)
2. Fetch slip of paper in mailbox with that number (e.g., in box 00)
3. Remember 3-digit number (e.g., 901) on paper, return paper to mailbox
4. Push “next” button (increment) on the instruction counter
5. Perform operation (e.g., 901) on slip of paper

Called the "Fetch-Decode-Execute" cycle:
- fetch instruction from memory, decode it (type, operands), and execute it
More details:

There is an LMC operator (different from the little man) that:

• clears and places slips of paper (instructions) in the mailboxes
• resets the instruction counter to 00
• wakes up the little man to start program execution

The inbox and outbox hold paper slips for interacting with users:

• each paper slip contains a three-digit number
• little man adds new slips to the outbox tray
• user adds new slips to the inbox tray
• little man always takes slips from bottom of inbox (slip there the longest)
• slips taken from the inbox tray are never put back by the little man

The calculator:

• stores 3 digit numbers (the “accumulator”)
• can add and subtract from the stored 3-digit number (result stored)
• if result of subtracting is negative, screen turns red (“negative flag”)
• if value is zero, screen turns green (“zero flag”)

Instructions:

• First digit of an instruction is the “operation code” (or “opcode” for short)
• Operations to add, subtract, store, load, input/output, etc.
• Rest of instruction is an “operand”, usually a mailbox number (“address”)
An example LMC program to add 2 user-supplied numbers:

Mailboxes:

00: 901  
01: 306  
02: 901  
03: 106  
04: 902
05: 000  
06: 000  
extc.

Inbox:

1st paper slip: 004
2nd paper slip: 003

Trace of the program run:

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>Result</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 00 | 901         | accumulator = 004 | opcode 9 is input/output 
      |              | operand 01 is input from inbox |
| 01 | 306         | mailbox 06 = 004  | opcode 3 is store accumulator value 
      |              | operand 06 is mailbox address |
| 02 | 901         | accumulator = 003 | similar to above |
| 03 | 106         | accumulator = 007 | opcode 1 is add mailbox value 
      |              | to accumulator value |
| 04 | 902         | outbox slip: 007  | opcode 9, operand 02 is output 
      |              | accumulator written to outbox slip |
| 05 | 000         |             | opcode 0 is halt – little man rests 
      |              | waits for operator |
Exercise: Write an LMC program to compute 42 + 11 + 9

Mailboxes:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00:</td>
<td>901</td>
<td>01:</td>
<td>309</td>
<td>02:</td>
<td>901</td>
</tr>
<tr>
<td>05:</td>
<td>110</td>
<td>06:</td>
<td>109</td>
<td>07:</td>
<td>902</td>
</tr>
<tr>
<td>10:</td>
<td>000</td>
<td>11:</td>
<td>000</td>
<td>12:</td>
<td>000</td>
</tr>
<tr>
<td>04:</td>
<td>901</td>
<td>09:</td>
<td>000</td>
<td>13:</td>
<td>000</td>
</tr>
<tr>
<td>14:</td>
<td>000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inbox:

1st paper slip: 009
2nd paper slip: 011
3rd paper slip: 042

Program trace:

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>Op</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
<td>read</td>
<td>accumulator = 009</td>
</tr>
<tr>
<td>01</td>
<td>309</td>
<td>store</td>
<td>mailbox 09 = 009</td>
</tr>
<tr>
<td>02</td>
<td>901</td>
<td>read</td>
<td>accumulator = 011</td>
</tr>
<tr>
<td>03</td>
<td>310</td>
<td>store</td>
<td>mailbox 10 = 011</td>
</tr>
<tr>
<td>04</td>
<td>901</td>
<td>read</td>
<td>accumulator = 042</td>
</tr>
<tr>
<td>05</td>
<td>110</td>
<td>add</td>
<td>accumulator = 053</td>
</tr>
<tr>
<td>06</td>
<td>109</td>
<td>add</td>
<td>accumulator = 062</td>
</tr>
<tr>
<td>07</td>
<td>902</td>
<td>write</td>
<td>outbox slip: 062</td>
</tr>
<tr>
<td>08</td>
<td>000</td>
<td>halt</td>
<td></td>
</tr>
</tbody>
</table>
# Full set of LMC opcodes

<table>
<thead>
<tr>
<th>opcode</th>
<th>mnemonic</th>
<th>description</th>
</tr>
</thead>
</table>
| 1xx    | ADD      | Add 3-digit value in mailbox \( xx \) to accumulator  
If result is 4-digits, only keep 3 (overflow) |
| 2xx    | SUB      | Subtract 3-digit value in mailbox \( xx \) from accumulator  
If result is negative, negative flag is set (led becomes red) |
| 3xx    | STA      | Store accumulator value into mailbox \( xx \)  
Accumulator unchanged but previous mailbox value discarded |
| 5xx    | LDA      | Load value in mailbox \( xx \) into accumulator  
Accumulator changed, mailbox unchanged |
| 6xx    | BRA      | Set the instruction counter to value \( xx \)  
Jumps (or branches) to different place in program |
| 7xx    | BRZ      | Set instruction counter to \( xx \) if zero-flag set  
Zero is considered positive |
| 8xx    | BRP      | Set instruction counter to \( xx \) if negative-flag not set  
Another conditional jump |
| 901    | INP      | Enter 3-digit value in inbox into accumulator  
Waits until value available, value discarded |
| 902    | OUT      | Copy 3-digit value in accumulator into outbox  
Accumulator value is unchanged |
| 000    | HLT      | Stop working / end of program |
Example program using a branch instruction

- Below we use mailboxes (after halt) to store constants
- Referred to as a data (DAT) “instruction”

Mailboxes:

00: 901 (INP) 01: 209 (SUB) 02: 806 (BRP) 03: 510 (LDA)
04: 902 (OUT) 05: 608 (BRA) 06: 511 (LDA) 07: 902 (OUT)
08: 000 (HLT) 09: 010 (DAT) 10: 001 (DAT) 11: 000 (DAT)

“Pseudocode”:

if (x > 10) // x is value in inbox
    return 0
else
    return 1

Inbox:

1st paper slip: 013

Program trace:

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>Op</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
<td>read</td>
<td>accumulator = 020</td>
</tr>
<tr>
<td>01</td>
<td>209</td>
<td>subtract</td>
<td>accumulator = 003</td>
</tr>
<tr>
<td>02</td>
<td>806</td>
<td>branch positive</td>
<td>pc = 06</td>
</tr>
<tr>
<td>06</td>
<td>511</td>
<td>load</td>
<td>accumulator = 000</td>
</tr>
<tr>
<td>07</td>
<td>902</td>
<td>write</td>
<td>outbox: 000</td>
</tr>
<tr>
<td>08</td>
<td>000</td>
<td>halt</td>
<td></td>
</tr>
</tbody>
</table>
LMC “Assembly-Like” Programs (Mnemonics and Labels)

- Instead of numeric opcodes, use the instruction mnemonics
- Can also use labels for instruction addresses (for jump locations and data)

Previous example using mnemonics and labels:

```
INP
SUB ten
BRP ifcase
LDA one
OUT
BRA done
ifcase LDA zero
OUT
done HLT
ten 010
one 001
zero 000
```

The trace (with same input of 013):

<table>
<thead>
<tr>
<th>Pc</th>
<th>Instruction</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>INP</td>
<td>accumulator = 020</td>
</tr>
<tr>
<td>01</td>
<td>SUB 10</td>
<td>accumulator = 003</td>
</tr>
<tr>
<td>02</td>
<td>BRP 06</td>
<td>pc = 06</td>
</tr>
<tr>
<td>06</td>
<td>LDA 11</td>
<td>accumulator = 000</td>
</tr>
<tr>
<td>07</td>
<td>OUT</td>
<td>outbox: 000</td>
</tr>
<tr>
<td>08</td>
<td>HLT</td>
<td></td>
</tr>
</tbody>
</table>
Exercise: Write an “assembly-like” (mnemonic) LMC version of:

```
while (x >= 0) { // assume x from inbox
    print(x); // output x via outbox
    x = x - 1; // decrement x
}
```

One solution:

```
INP
cond BRP body
    BRA done
body OUT
    SUB one
    BRA cond
done HLT
one DAT 001
```

LMC Simulator: [https://www.101computing.net/LMC/](https://www.101computing.net/LMC/)

- link given in homework
- might help test and debug your homework, etc