Admin

Midterm 1
- Practice problems posted
- Very light coverage of numbers with different bases (I wouldn’t put Q7 on the midterm)

Assignment 2 grading
Assignment 3
Assignment 4

Examples from this lecture

https://cs.pomona.edu/classes/cs54/examples/cs52machine

Admin

Extra mentor hours:
- Monday, 8-10pm (Gabriel)
- Tuesday, 8:15-9:30 (Will)

My Wednesday office hours:
- 9:30-10:30am (No hours 10:30am-12)
Computer internals simplified

CPU
What does it stand for?
What does it do?

RAM
What does it stand for?
What does it do?

CPU
(Central Processing Unit, aka “the processor”)

RAM
(Random Access Memory, aka “memory” or “main memory”)

Does all the work!
Temporary storage

Computer internals simplified

"the computer"

Why do we need a hard drive?

Computer internals

Computer internals simplified

CPU

RAM

hard drive

"the computer"
Computer internals simplified

"the computer"
- CPU
- RAM
- hard drive

"the computer"
- Persistent memory
- RAM only stores data while it has power

Computer simplified

CPU
RAM
hard drive
media drive
input devices
display

Inside the CPU

CPU
- processor: does the work
- registers: local, fast memory slots

Why all these levels of memory?

Memory speed

<table>
<thead>
<tr>
<th>operation</th>
<th>access time</th>
<th>times slower than register access</th>
<th>for comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>register</td>
<td>0.3 ns</td>
<td>1</td>
<td>1 s</td>
</tr>
<tr>
<td>RAM</td>
<td>120 ns</td>
<td>400</td>
<td>6 min</td>
</tr>
<tr>
<td>Hard disk</td>
<td>1 ms</td>
<td>~million</td>
<td>1 month</td>
</tr>
<tr>
<td>box, onedrive,</td>
<td>0.4s</td>
<td>~billion</td>
<td>30 years</td>
</tr>
</tbody>
</table>
What is a byte?

My laptop has 32GB (gigabytes) of memory. How many bits is that?

Memory sizes

<table>
<thead>
<tr>
<th>Size</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8</td>
</tr>
<tr>
<td>kilobyte (KB)</td>
<td>$2^{10}$ bytes = 8,000</td>
</tr>
<tr>
<td>megabyte (MB)</td>
<td>$2^{20}$ = 8 million</td>
</tr>
<tr>
<td>gigabyte (GB)</td>
<td>$2^{30}$ = 8 billion</td>
</tr>
</tbody>
</table>

~256 billion bits!
Memory is byte addressable.

Memory is organized into “words”, which is the most common functional unit.

Most modern computers use 32-bit (4 byte) or 64-bit (8 byte) words.

We'll use 16-bit words for our model (the CS52 machine).
CS52 machine

CPU

instruction counter (location in memory of the next instruction in memory)
holds the value 0 (read only)
- general purpose - read/write

r0
r1
r2
r3

CS52 machine instructions

What types of operations might we want to do (think really basic)?

CS52 machine code

Four main types of instructions
1. math
2. branch/conditionals
3. memory
4. control the machine (e.g. stop it)

In executing a program, the CS52 Machine follows:
- The machine fetches the value at mem[ic] for use as an instruction.
- The machine increments the value in ic by 2.
- The machine decodes and carries out the instruction.
instruction name arguments

add, sub, and, or, xor

RRR or RRS

25

instruction name arguments

add, sub, and, or, xor

RRR or RRS

26

instruction/operation name
(always three characters)

27

operation arguments
R = register (e.g. r0)
S = signed number (byte)

28

1st R: register where the answer will go
2nd R: register of first operand
3rd S/R: register/value of second operand
add r1 r2 r3

What does this do?

1st R: register where the answer will go
2nd R: register of first operand
3rd S/R: register/value of second operand

r1 = r2 + r3
Add contents of registers r2 and r3 and store the result in r1

1st R: register where the answer will go
2nd R: register of first operand
3rd S/R: register/value of second operand

add r2 r1 10

What does this do?

r2 = r1 + 10
Add 10 to the contents of register r1 and store in r2

1st R: register where the answer will go
2nd R: register of first operand
3rd S/R: register/value of second operand
add r1 r0 8
neg r2 r1
sub r2 r1 r2

What number is in r2?

1st R: register where the answer will go
2nd R: register of first operand
3rd S/R: register/value of second operand

add r1 r0 8
neg r2 r1
sub r2 r1 r2

r1 = 8
r2 = -8, r1 = 8
r2 = 16

1st R: register where the answer will go
2nd R: register of first operand
3rd S/R: register/value of second operand

Accessing memory

sto = save data in register TO memory
loa = put data FROM memory into a register

sto r1 r2 ; store the contents of r1 to mem[r2]
loa r1 r2 ; get data from mem[r2] and put into r1

Accessing memory

sto = save data in register TO memory
loa = put data FROM memory into a register

Special cases:
- saving TO (sto) address 0 prints
- reading from (loa) address 0 gets input from user
Basic structure of CS52 program

; great comments at the top!

; instruction1    ; comment
; instruction2    ; comment
...
; hlt

whitespace before operations/instructions

Running the CS52 machine

Look at subtract.a52
- load two numbers from the user
- subtract
- print the result

CS52 simulator

Different windows
- Memory (left)
- Instruction execution (right)
- Registers
- I/O and running program

1st R: first register for comparison
2nd R: second register in comparison
3rd B: label
beq r3 r0 done

1st R: first register for comparison
2nd R: second register in comparison
3rd B: label

What does this do?

If r3 = 0, branch to the label “done” if not (else) ic is incremented as normal to the next instruction

ble r2 r3 done

1st R: first register for comparison
2nd R: second register in comparison
3rd B: label

What does this do?

If r2 <= r3, branch to the label done
- Conditionals
- Loops
- Change the order that instructions are executed

Basic structure of CS52 program

```assembly
; great comments at the top!
/
  instruction1 ; comment
  instruction2 ; comment
...
label1
  instruction ; comment
  instruction ; comment
label2
...
  bit
  end

- whitespace before operations/instructions
- labels go here
```

CS52 machine execution

A program is simply a sequence of instructions stored in a block of contiguous
words in the machine’s memory. In executing a program, the CS52 Machine
follows a simple loop:

- The machine fetches the value at mem[ic] for use as an instruction.
- The machine increments the value in ic by 2.
- The machine decodes and carries out the instruction.

More CS52 examples

Look at max_simple.a52
- Get two values from the user
- Compare them
- Use a branch to distinguish between the two cases
  - Goal is to get largest value in r3
  - print largest value