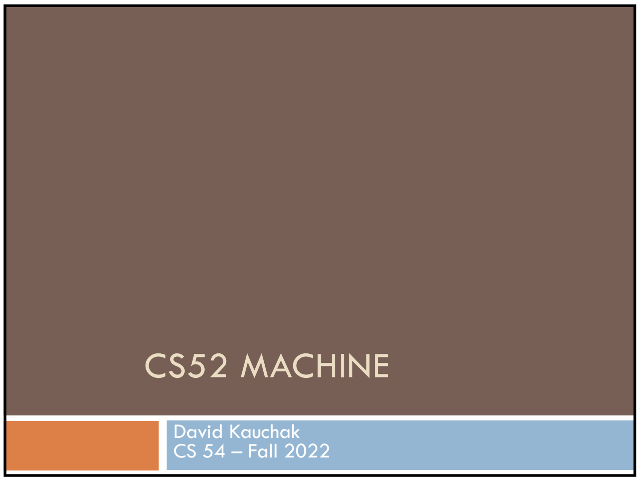
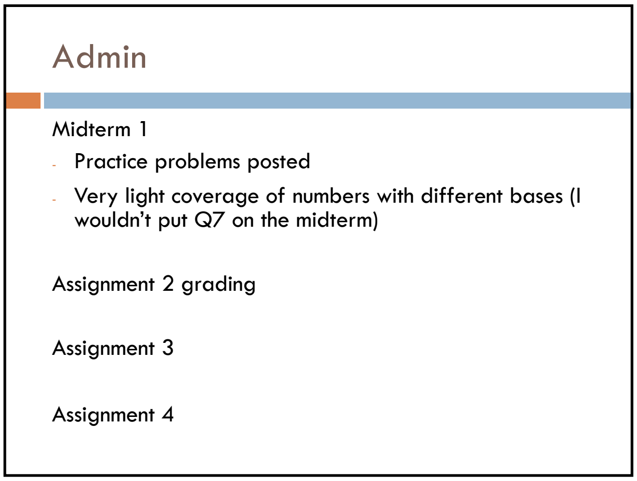


# CS52 MACHINE

David Kauchak
CS 54 – Fall 2022

1



## Admin

Midterm 1

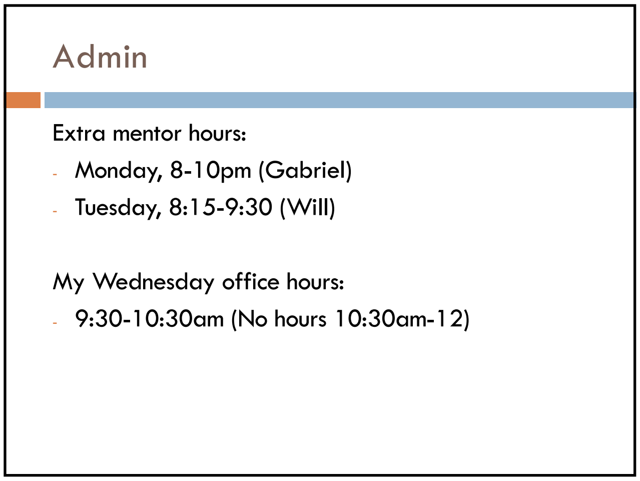
- Practice problems posted
- Very light coverage of numbers with different bases (I wouldn't put Q7 on the midterm)

Assignment 2 grading

Assignment 3

Assignment 4

2



## Admin

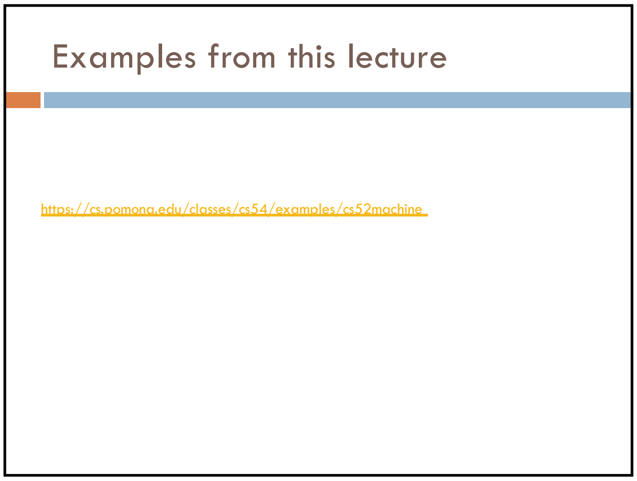
Extra mentor hours:

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

My Wednesday office hours:

- 9:30-10:30am (No hours 10:30am-12)

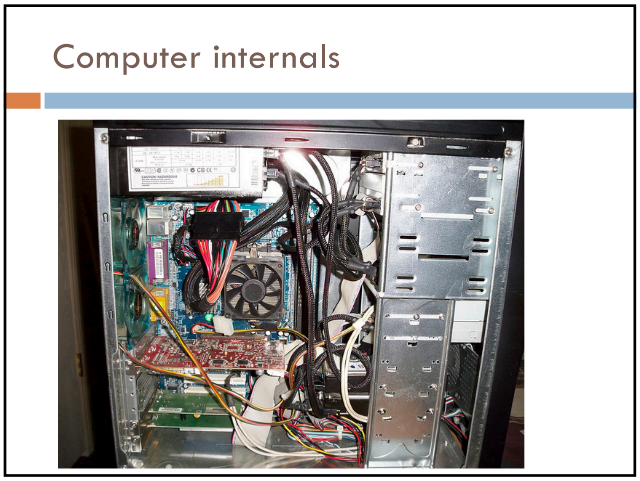
3



## Examples from this lecture

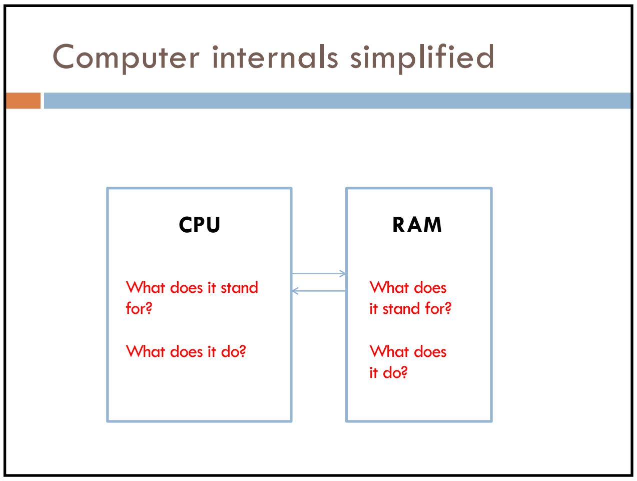
https://cs.pomona.edu/classes/cs54/examples/cs52machine

4



## Computer internals

5



## Computer internals simplified

**CPU**

What does it stand for?

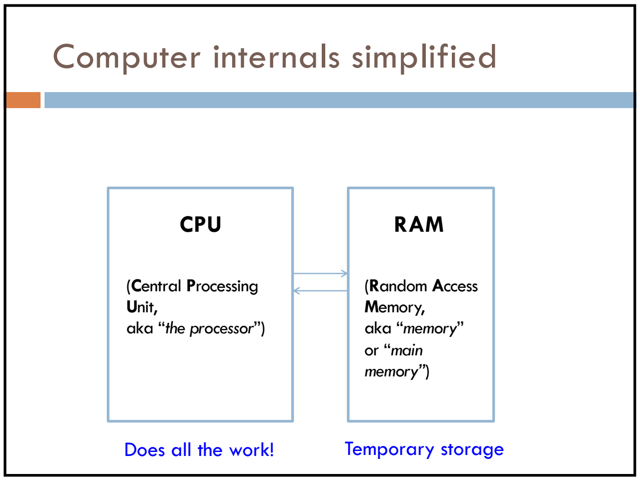
What does it do?

**RAM**

What does it stand for?

What does it do?

6



## Computer internals simplified

**CPU**

(**C**entral **P**rocessing **U**nit,
aka "*the processor*")

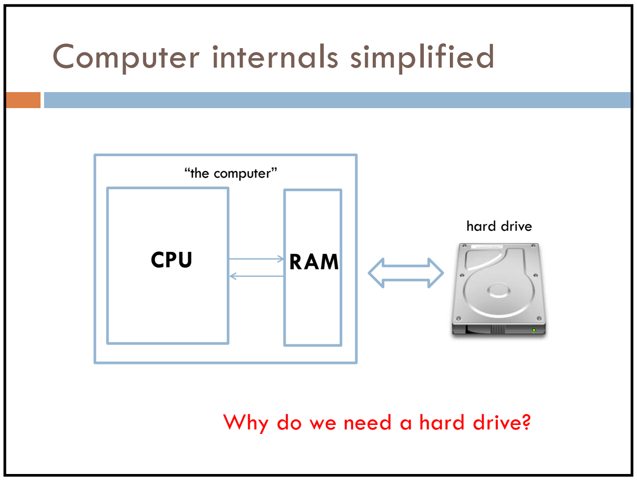
**RAM**

(**R**andom **A**ccess **M**emory,
aka "*memory*"
or "*main memory*")

Does all the work!

Temporary storage

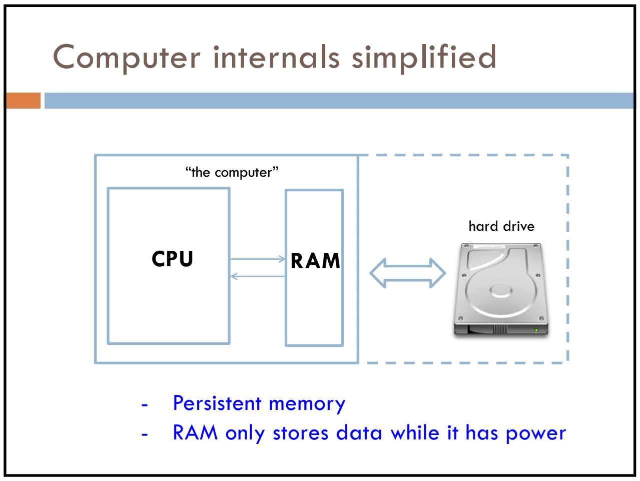
7



## Computer internals simplified

Why do we need a hard drive?

8



## Computer internals simplified

"the computer"

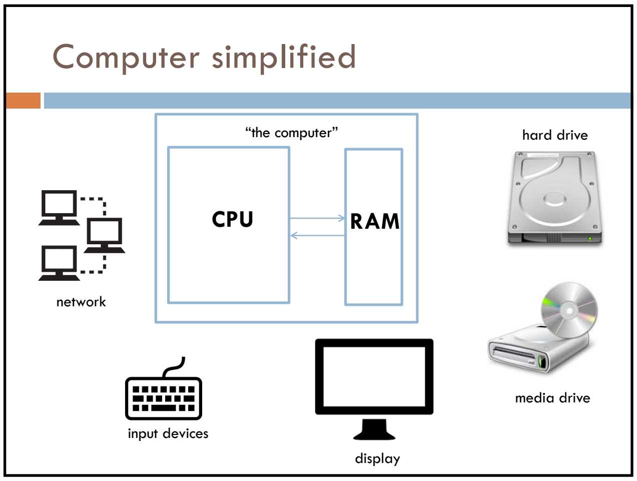
CPU

RAM

hard drive

- Persistent memory
- RAM only stores data while it has power

9



## Computer simplified

"the computer"

CPU

RAM

hard drive

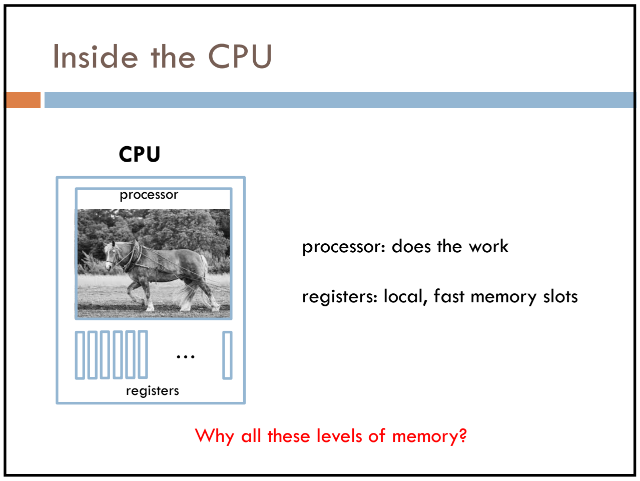
network

media drive

input devices

display

10



## Inside the CPU

CPU

processor

…

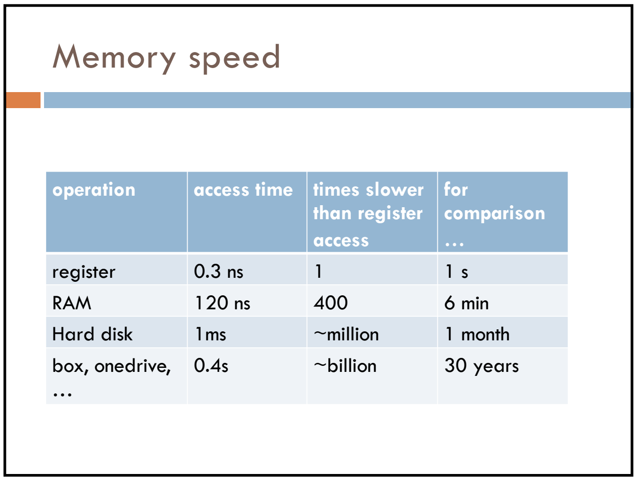
registers

processor: does the work

registers: local, fast memory slots

Why all these levels of memory?

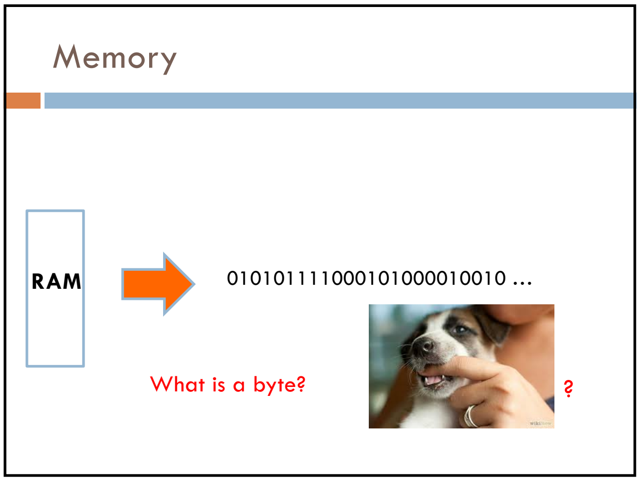
11



## Memory speed

| operation | access time | times slower than register access | for comparison … |
|---|---|---|---|
| register | 0.3 ns | 1 | 1 s |
| RAM | 120 ns | 400 | 6 min |
| Hard disk | 1ms | ~million | 1 month |
| box, onedrive, … | 0.4s | ~billion | 30 years |

12



## Memory

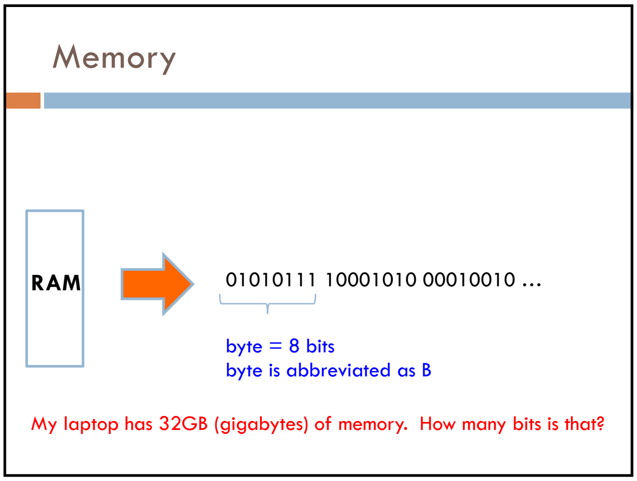
RAM

0101011110001010000010010 …

What is a byte?

?

13



## Memory

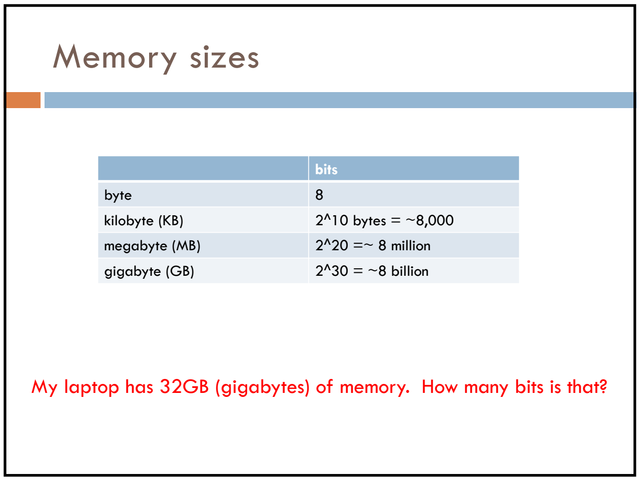
RAM

01010111 10001010 00010010 …

byte = 8 bits
byte is abbreviated as B

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

14

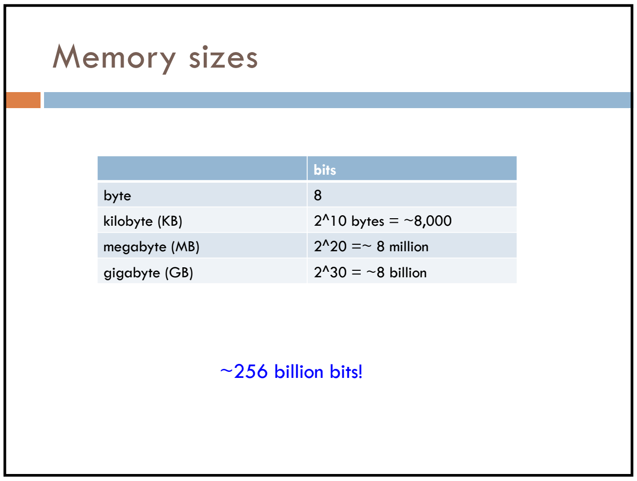


## Memory sizes

| | bits |
|---|---|
| byte | 8 |
| kilobyte (KB) | 2^10 bytes = ~8,000 |
| megabyte (MB) | 2^20 =~ 8 million |
| gigabyte (GB) | 2^30 = ~8 billion |

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

15

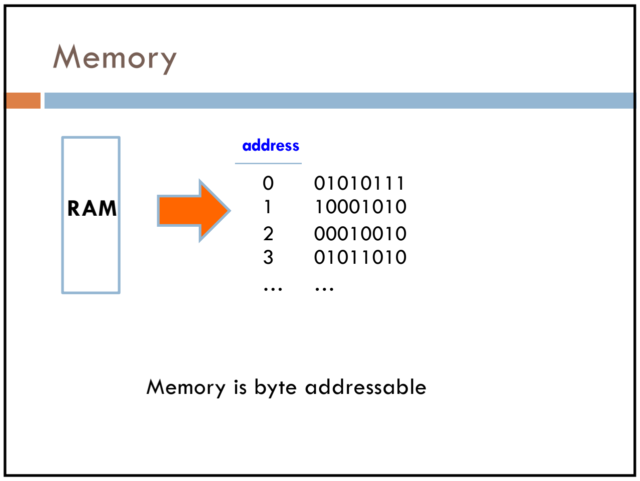


## Memory sizes

| | bits |
|---|---|
| byte | 8 |
| kilobyte (KB) | 2^10 bytes = ~8,000 |
| megabyte (MB) | 2^20 =~ 8 million |
| gigabyte (GB) | 2^30 = ~8 billion |

~256 billion bits!

16

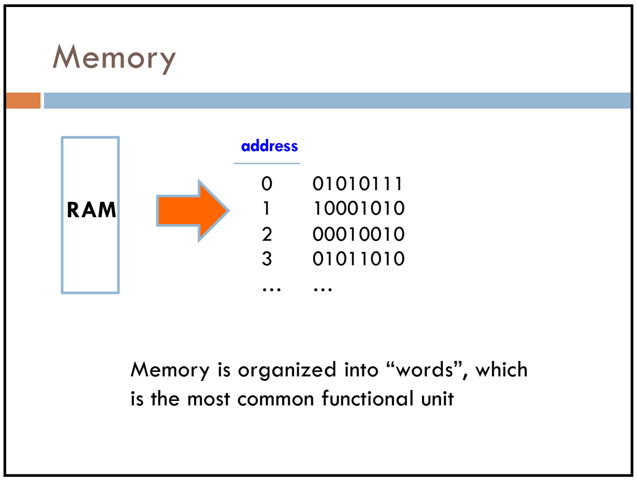


## Memory

| address |  |
|---|---|
| 0 | 01010111 |
| 1 | 10001010 |
| 2 | 00010010 |
| 3 | 01011010 |
| … | … |

Memory is byte addressable

17

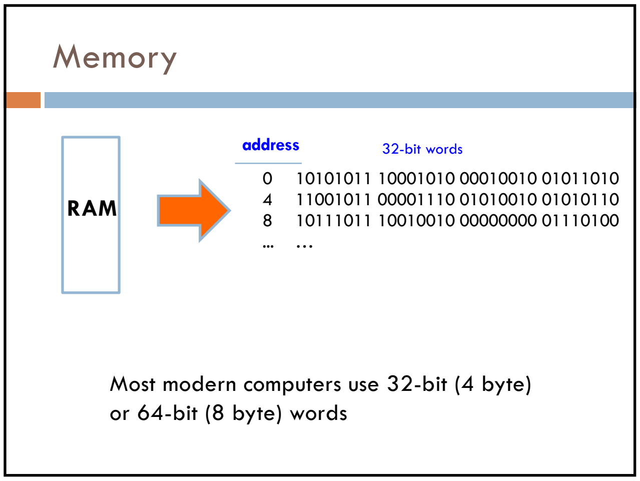


## Memory

| address |  |
|---|---|
| 0 | 01010111 |
| 1 | 10001010 |
| 2 | 00010010 |
| 3 | 01011010 |
| … | … |

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

18

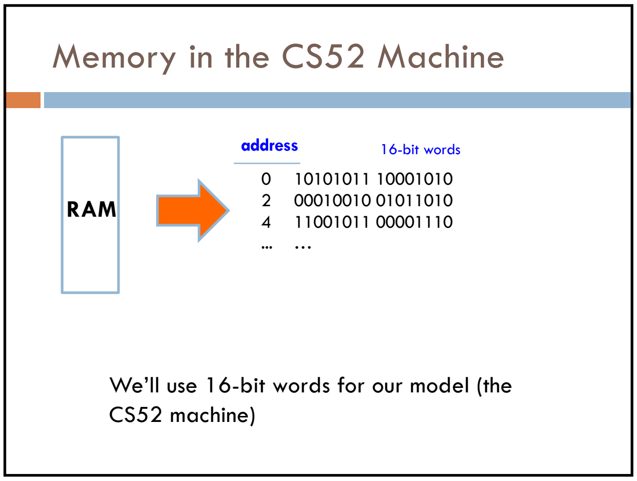


## Memory

| address | 32-bit words |
|---|---|
| 0 | 10101011 10001010 00010010 01011010 |
| 4 | 11001011 00001110 01010010 01010110 |
| 8 | 10111011 10010010 00000000 01110100 |
| … | … |

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

19



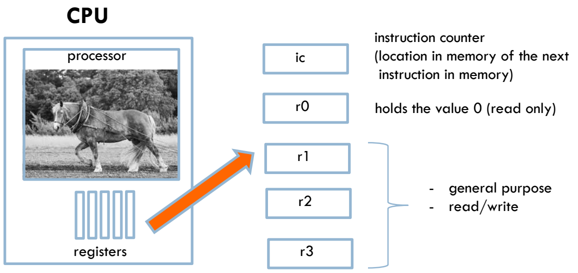
## Memory in the CS52 Machine

| address | 16-bit words |
|---|---|
| 0 | 10101011 10001010 |
| 2 | 00010010 01011010 |
| 4 | 11001011 00001110 |
| … | … |

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

20

## CS52 machine



instruction counter (location in memory of the next instruction in memory)

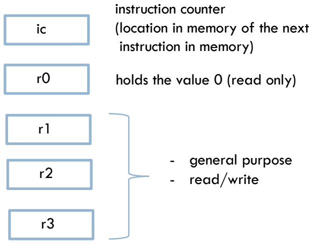
holds the value 0 (read only)

- general purpose
- read/write

21

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.



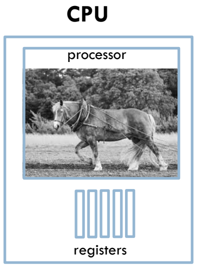
instruction counter (location in memory of the next instruction in memory)

holds the value 0 (read only)

- general purpose
- read/write

22

## CS52 machine instructions



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

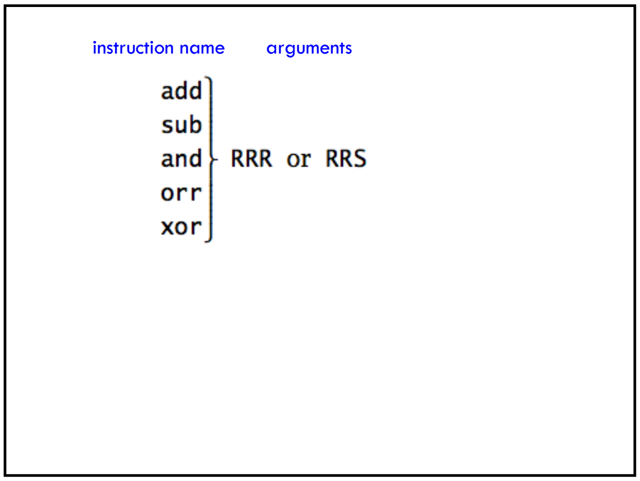
23

## CS52 machine code

Four main types of instructions

1. math
2. branch/conditionals
3. memory
4. control the machine (e.g. stop it)

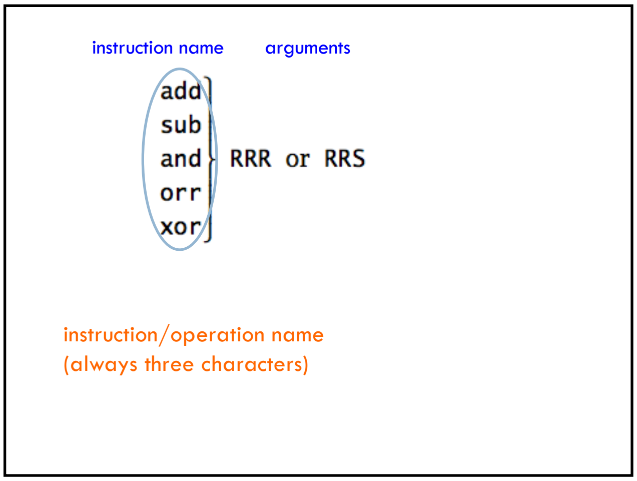
24



instruction name arguments

```
add ⎫
sub ⎪
and ⎬ RRR or RRS
orr ⎪
xor ⎭
```

25

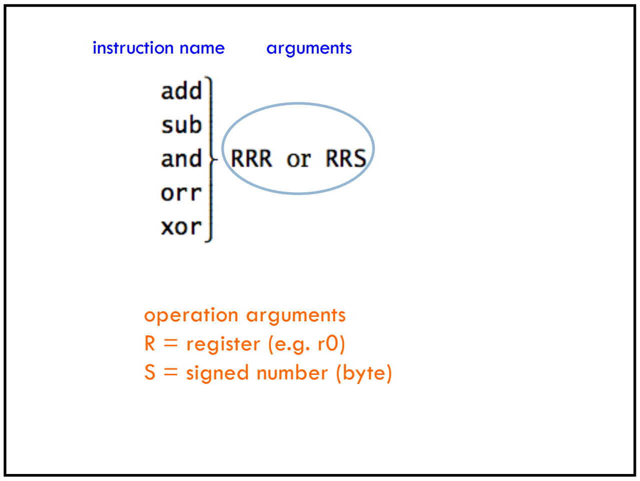


instruction name arguments

```
add ⎫
sub ⎪
and ⎬ RRR or RRS
orr ⎪
xor ⎭
```

instruction/operation name
(always three characters)

26

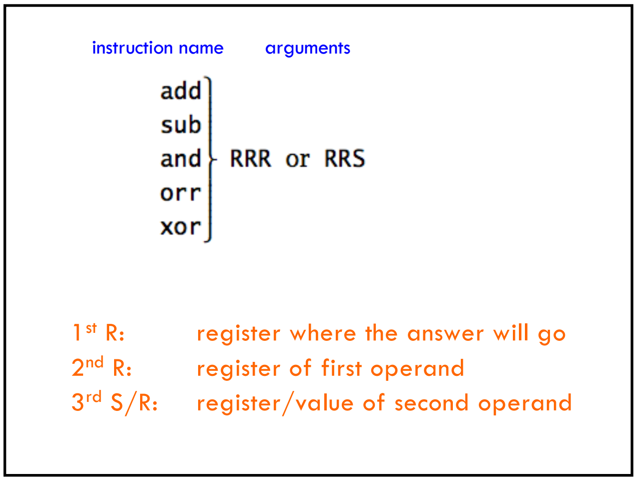


instruction name arguments

```
add ⎫
sub ⎪
and ⎬ RRR or RRS
orr ⎪
xor ⎭
```

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

27

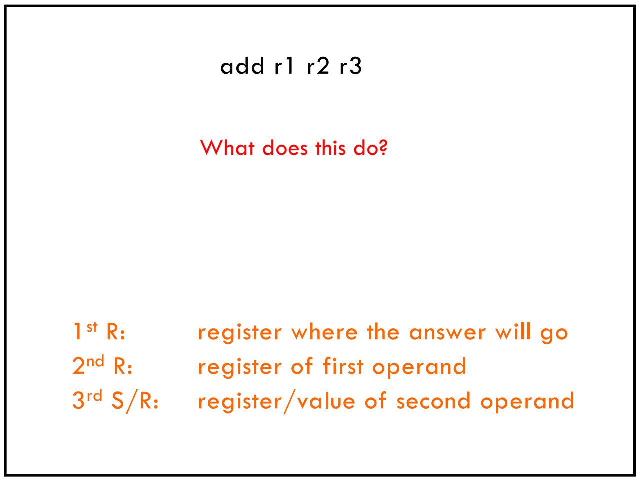


instruction name arguments

```
add ⎫
sub ⎪
and ⎬ RRR or RRS
orr ⎪
xor ⎭
```

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

28

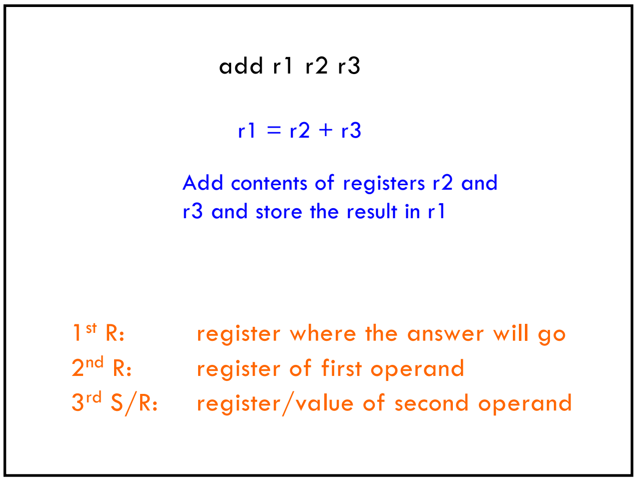


add r1 r2 r3

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

29



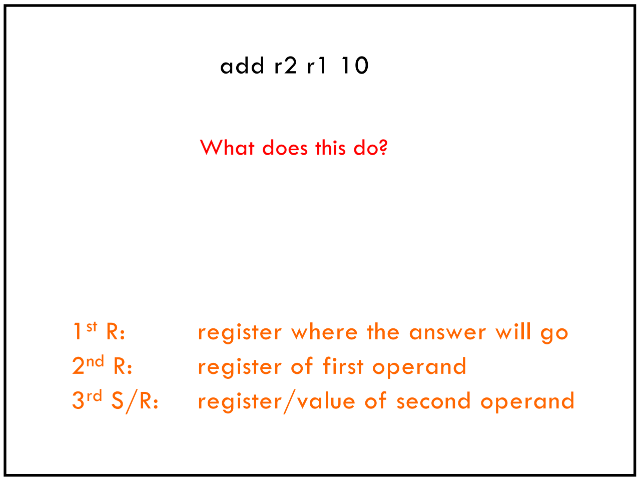
add r1 r2 r3

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

30

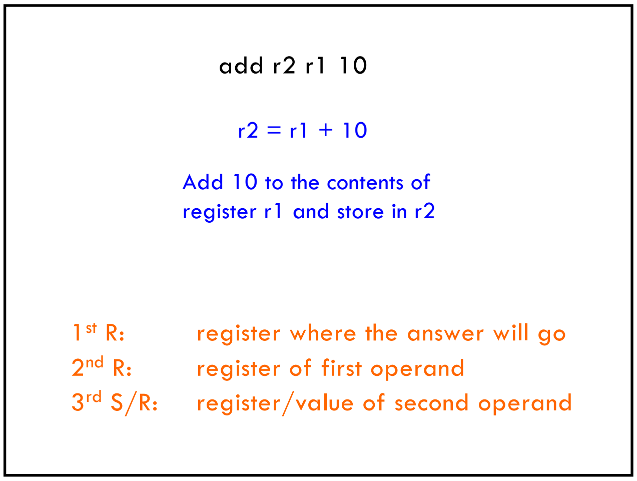


add r2 r1 10

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

31



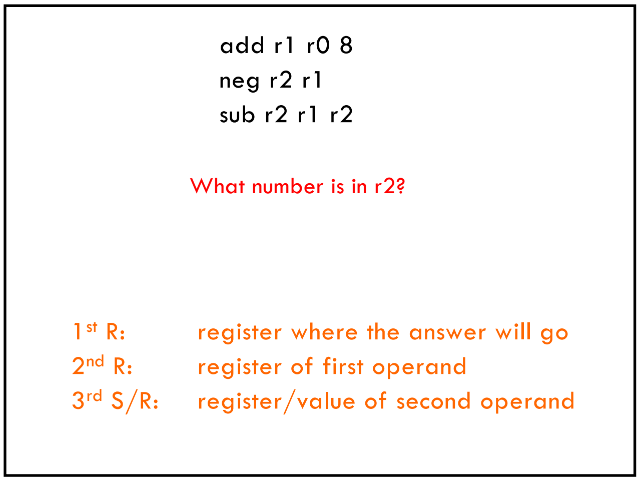
add r2 r1 10

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

32

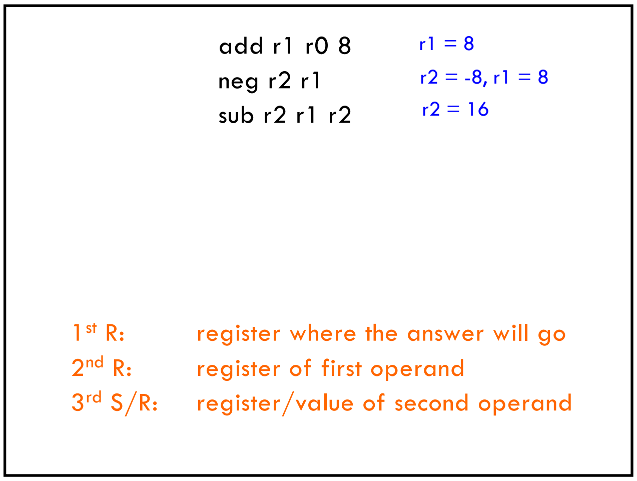


```
add r1 r0 8
neg r2 r1
sub r2 r1 r2
```

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

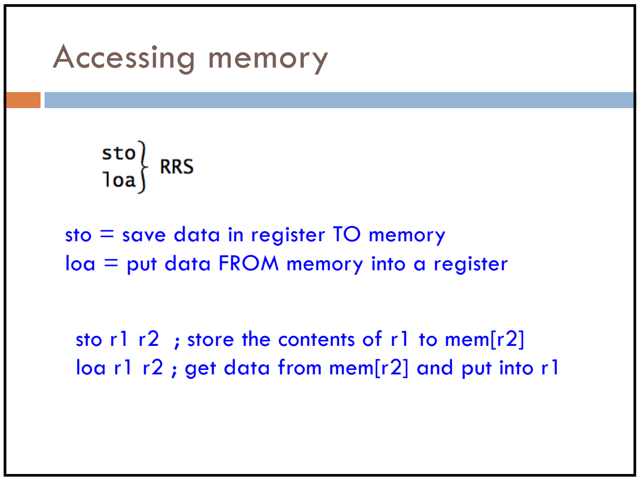
33



```
add r1 r0 8      r1 = 8
neg r2 r1        r2 = -8, r1 = 8
sub r2 r1 r2     r2 = 16
```

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

34



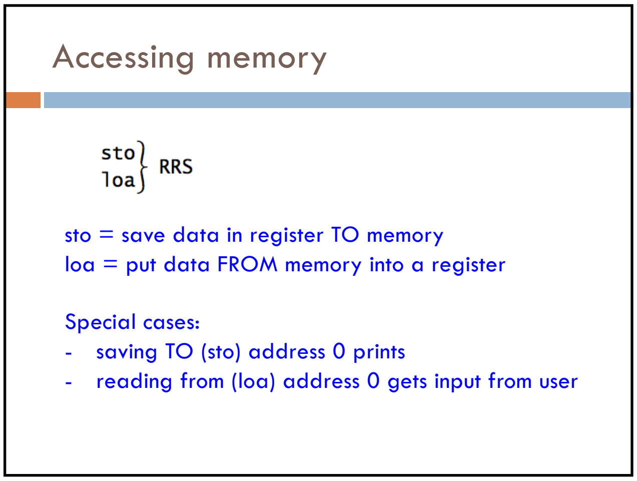
## Accessing memory

```
sto }
loa } RRS
```

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
```

35



## Accessing memory

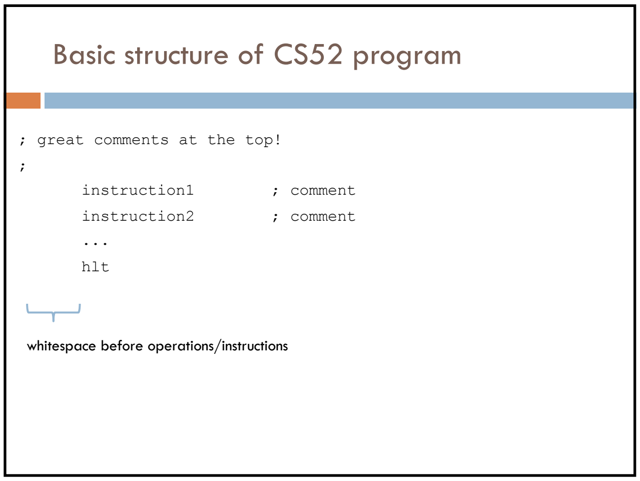
```
sto }
loa } RRS
```

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

36

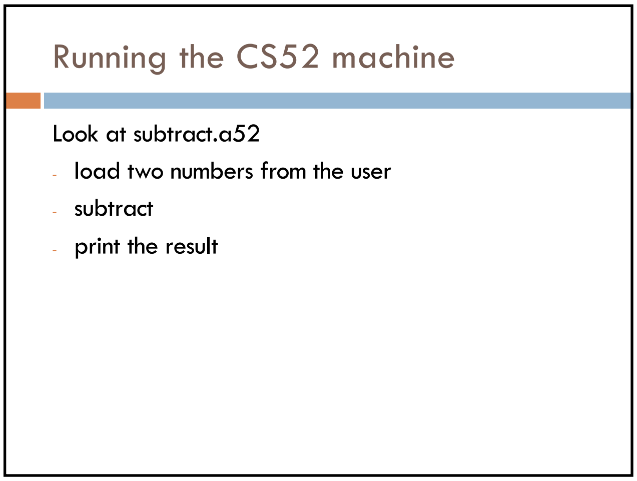


## Basic structure of CS52 program

```
; great comments at the top!
;
        instruction1         ; comment
        instruction2         ; comment
        ...
        hlt
```

whitespace before operations/instructions

37

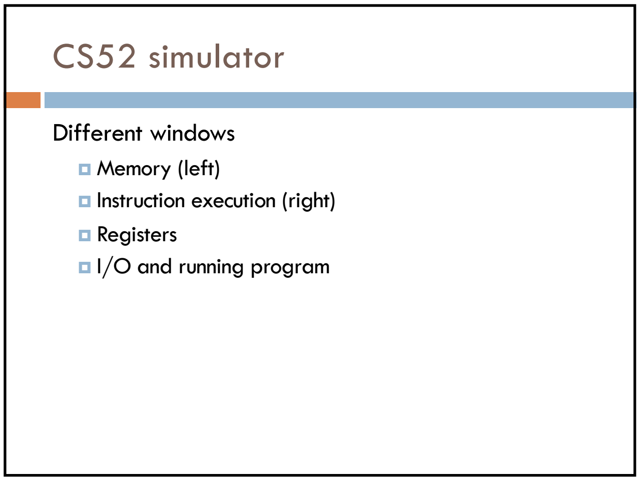


## Running the CS52 machine

Look at subtract.a52

- load two numbers from the user
- subtract
- print the result

38

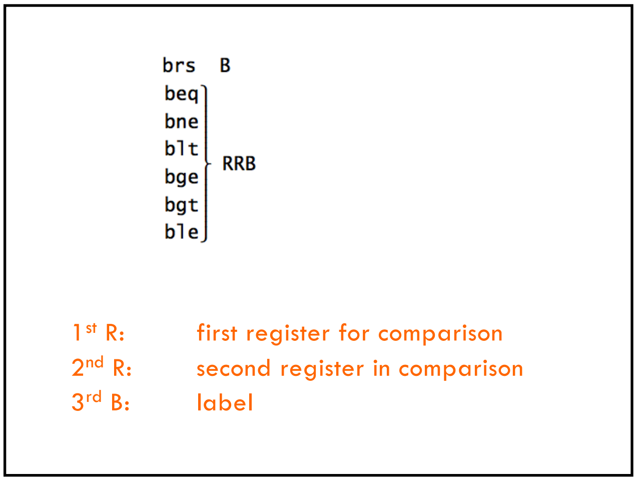


## CS52 simulator

Different windows

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

39



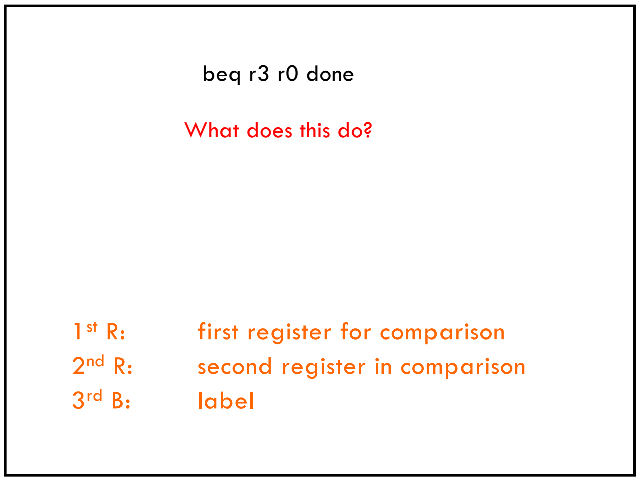
```
brs  B
beq ⎫
bne ⎪
blt ⎬ RRB
bge ⎪
bgt ⎪
ble ⎭
```

1st R: first register for comparison

2nd R: second register in comparison

3rd B: label

40

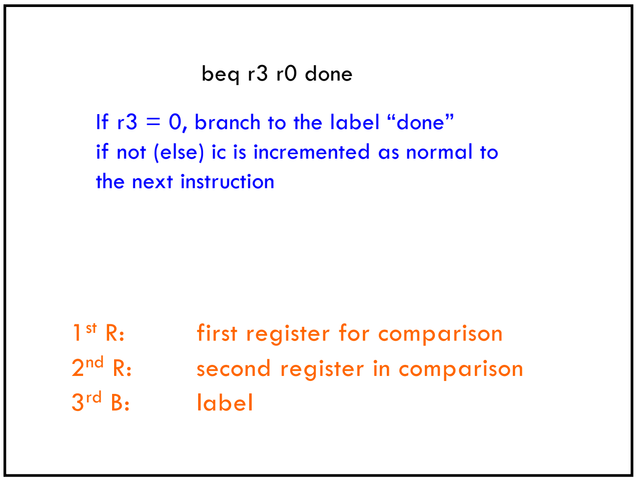


beq r3 r0 done

What does this do?

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

41

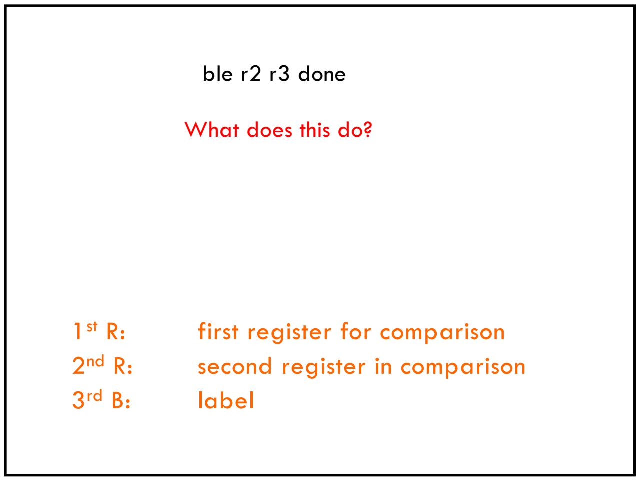


beq r3 r0 done

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

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

42

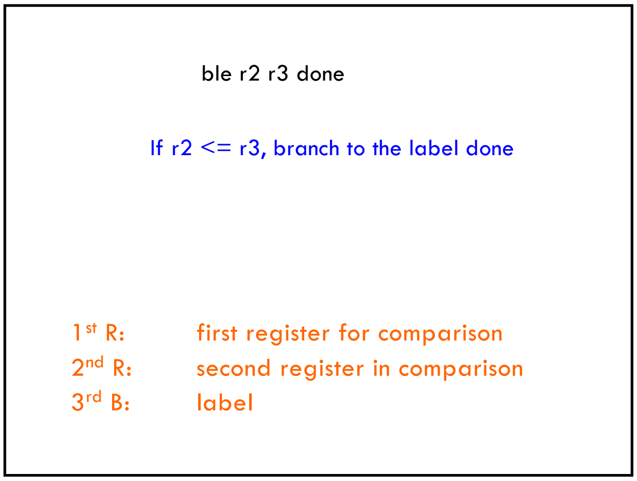


ble r2 r3 done

What does this do?

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

43



ble r2 r3 done

If r2 <= r3, branch to the label done

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

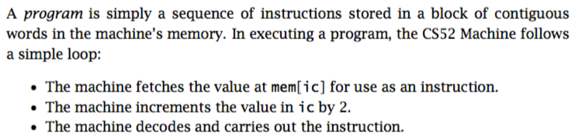
44

```
brs   B
beq ┐
bne │
blt │
bge ├ RRB
bgt │
ble ┘
```

- Conditionals
- Loops
- Change the order that instructions are executed

45

## 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.

46

## Basic structure of CS52 program

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

- whitespace before operations/instructions
- labels go here

47

## 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

48