CS 3650 – Computer Systems
Spring 2024
Peter Desnoyers

Lecture 2, Thur Jan 11 2024

Logistics

midterm exam Feb 29 (thur)

assignments: out Friday,
due Thursday

1+
handwritten notes

4 projects:
- → shell
- → file system
- → 2 others

Final exam ← 1 letter-sized sheet

The CPU

8 bits

1 0 1 1 0 0 1 1

signed

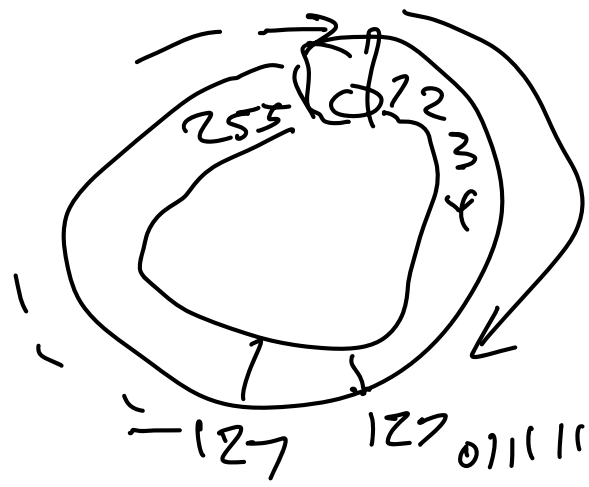
unsigned

$-128 \ldots 0 \ldots 127$

$0 \ldots 256$

0

127



-127?

1111 1111

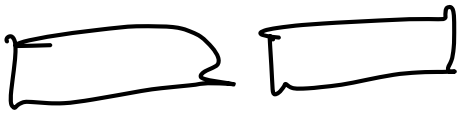
+1

0000 0000

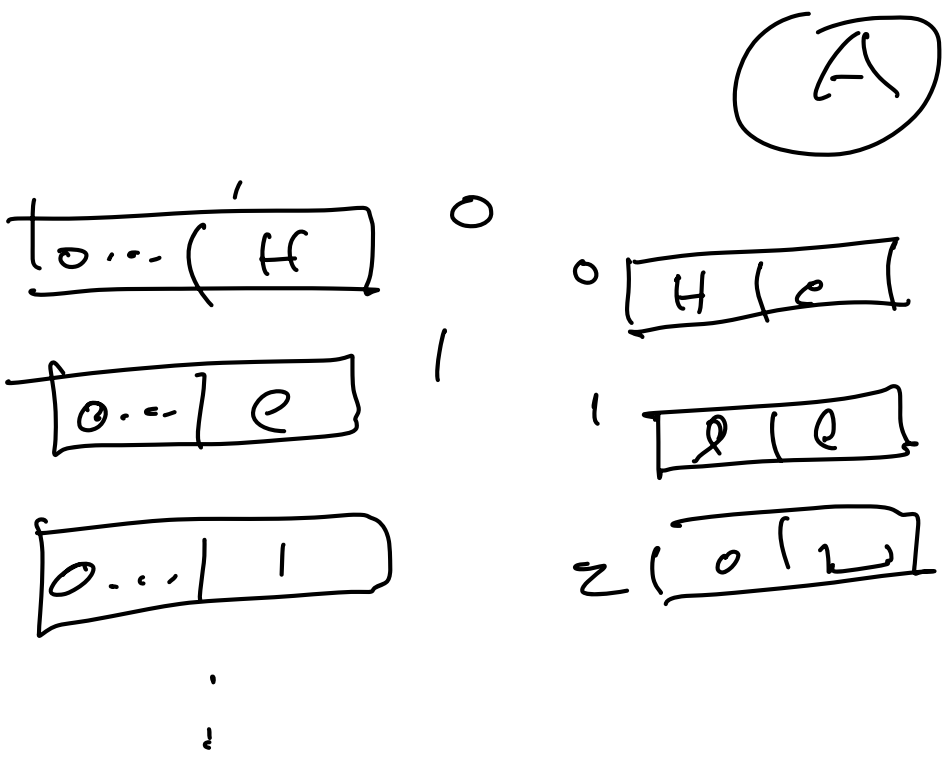
129

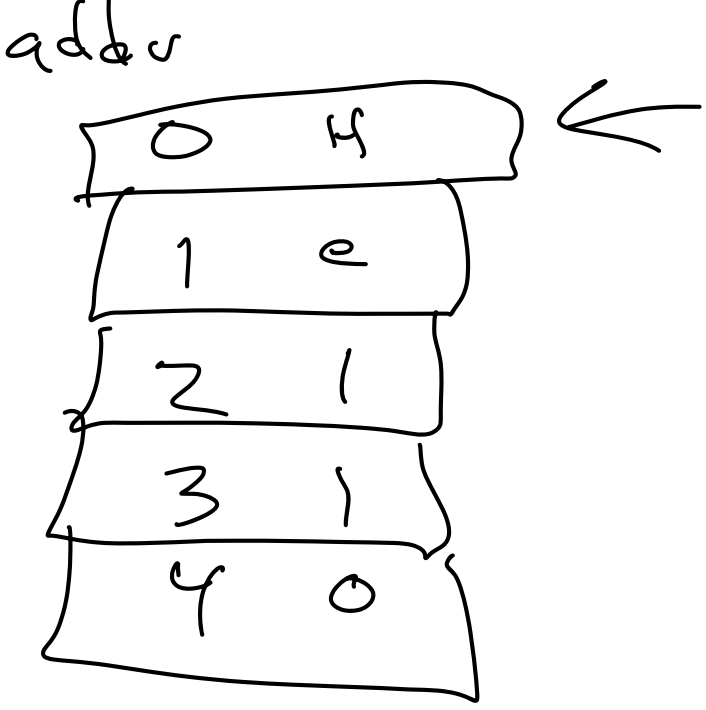
Bytes, words

(A) uint8 memory [64*1024]

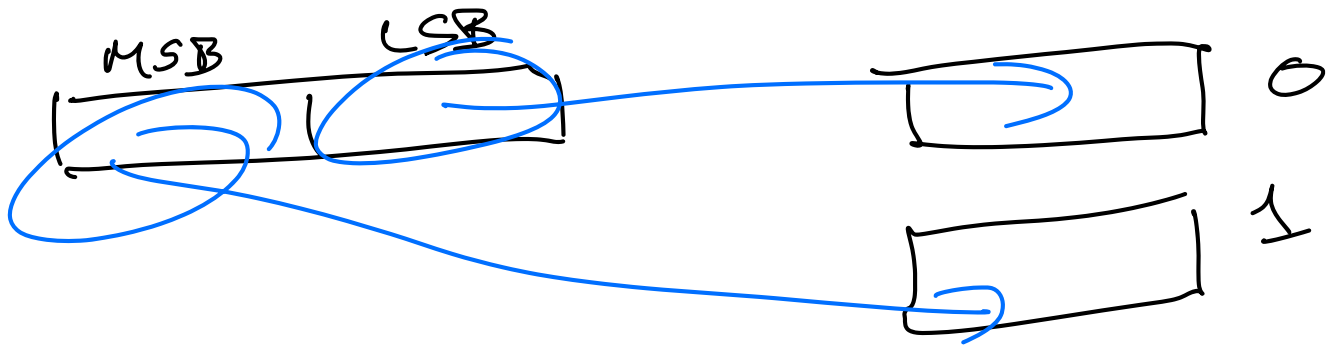


(B) uint16 memory [64*1024]





Larger words



CPU:

registers + instructions

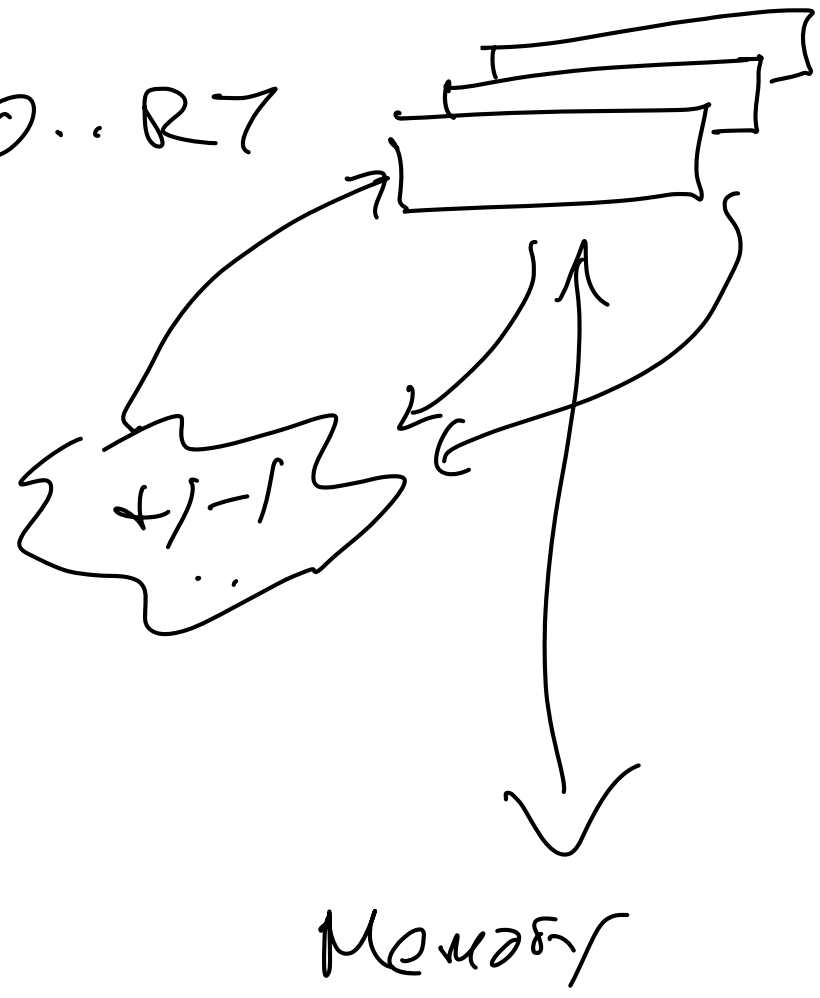
CPU registers (temp. variables)

8 16-bit registers

R0 .. R7

Program counter: PC

stack pointer: SP



how does it work ?

read2 (addr)
return mem[addr]
+ mem[addr+1] << 8

loop:

uint16 this_insn = read2 (PC)

← read 16 bits

if this_insn = X

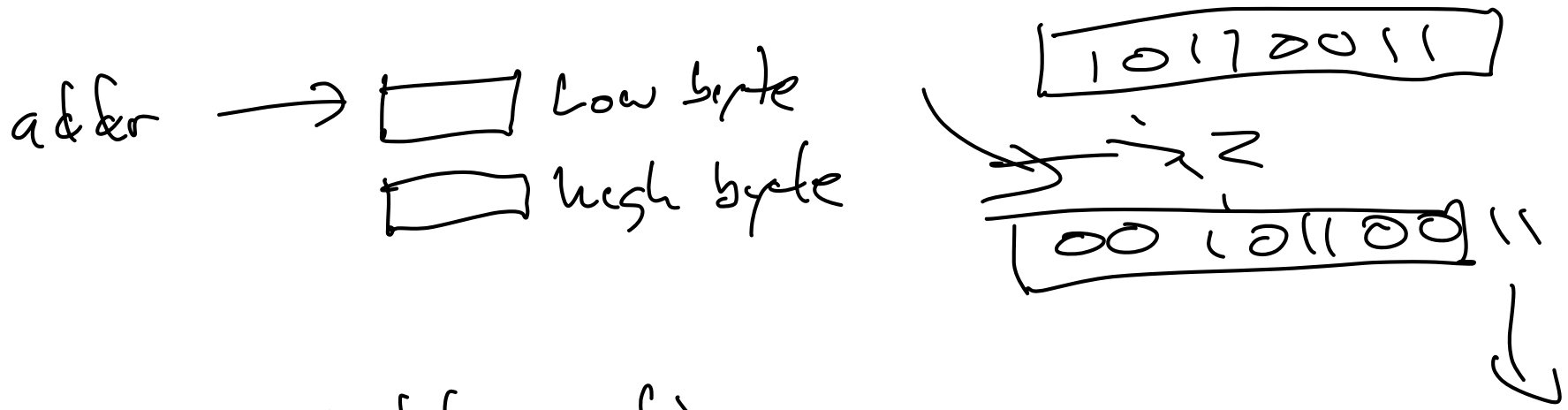
. . . .

else if = Y

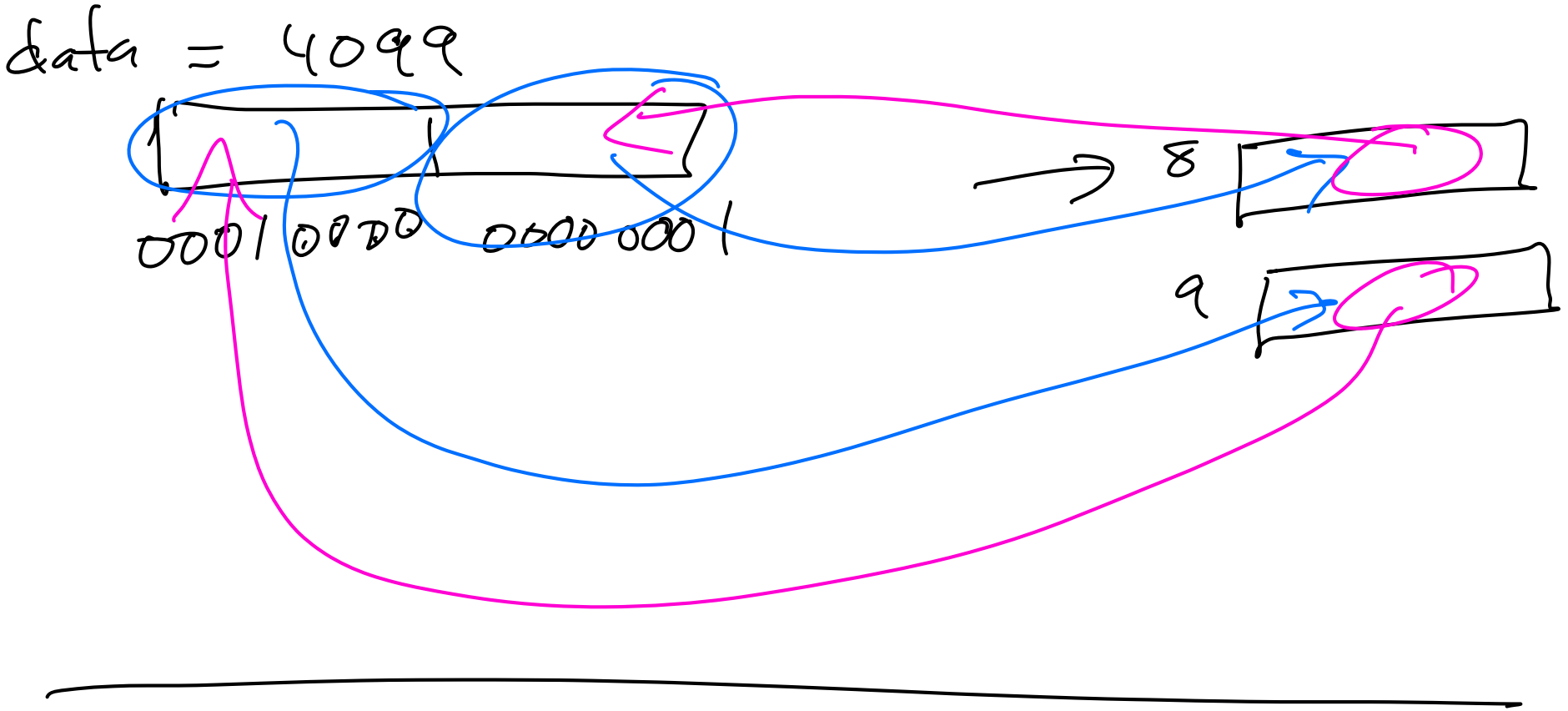
. . . ← set PC = PC+2

. .

```
read2(addr)
    return (mem[addr+1] << 8) | mem[addr]
```

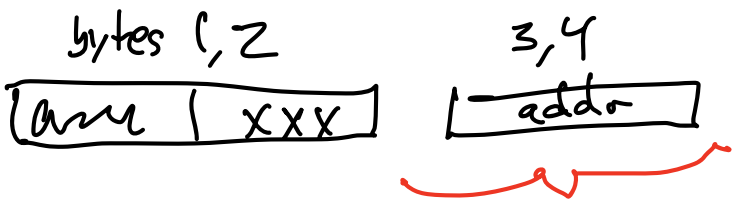



```
store2(addr, val)
    mem[addr] = data & 0xFF
    mem[addr+1] = data >> 8
```

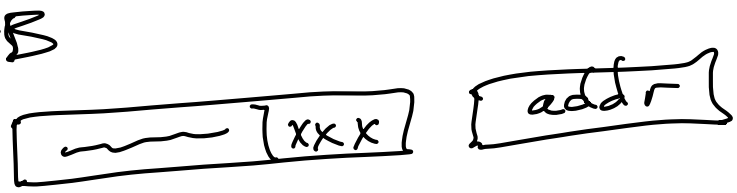

&data = 4099
00010000 00000001
8
9

Getting data into registers

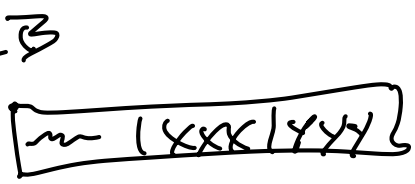
SET Rxxx = <value>



LOAD Rxxx ← <addr>

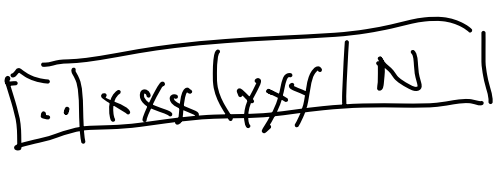


LOAD Rxxx ← *(Ryyy)



LOAD R2 ← *(R3)

LOAD Rxxx ← *(Ryyy+N)



Storing it again

STORE (. . . . .)

Does something with it

ADD Rxxx + Ryyy → Rzzz

SUB Rx - Ry → Rz

AND & OR (. . . - ~

CMP Rx - Ry ↘ ϕ

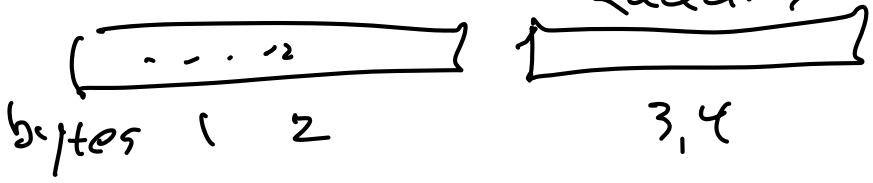
Flags

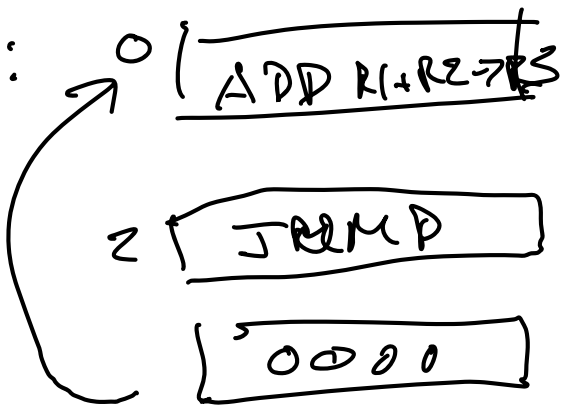
Z : zero

N : negative

Control flow - JUMP

JUMP <addr>





JUMP_Z, NZ zero, not zero

LT, GT less than (neg)

Greater than . .

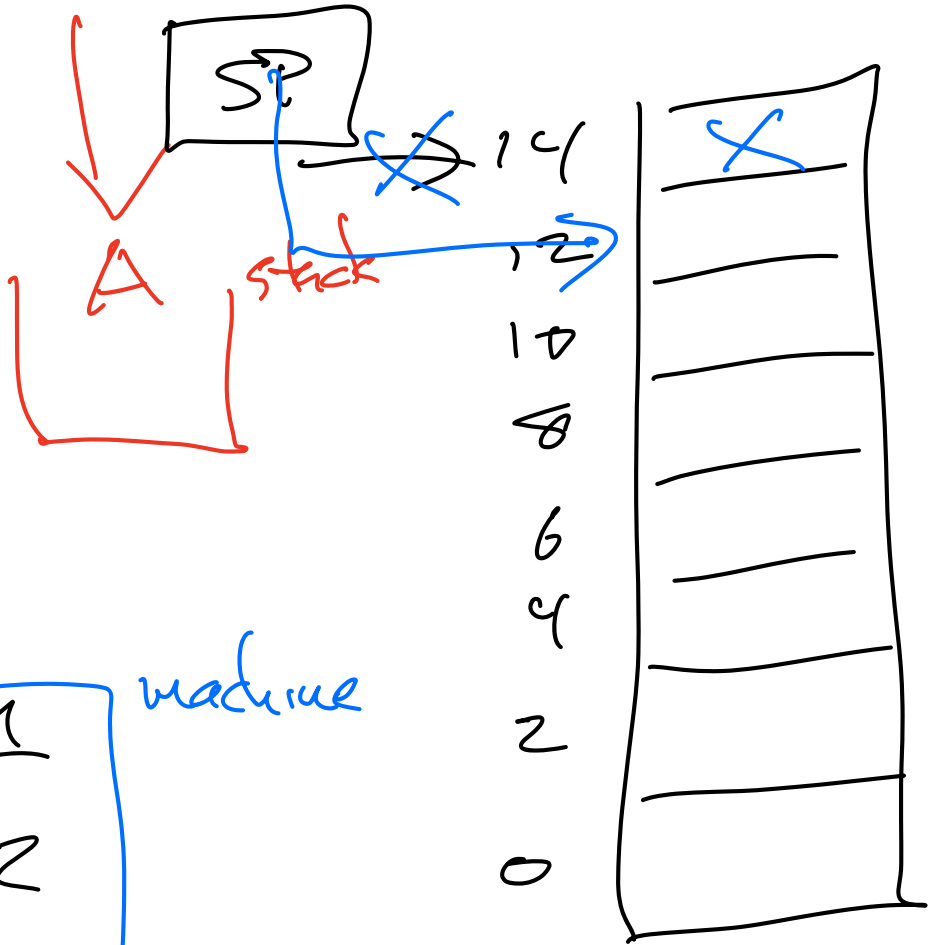
set PC (for next instruction)

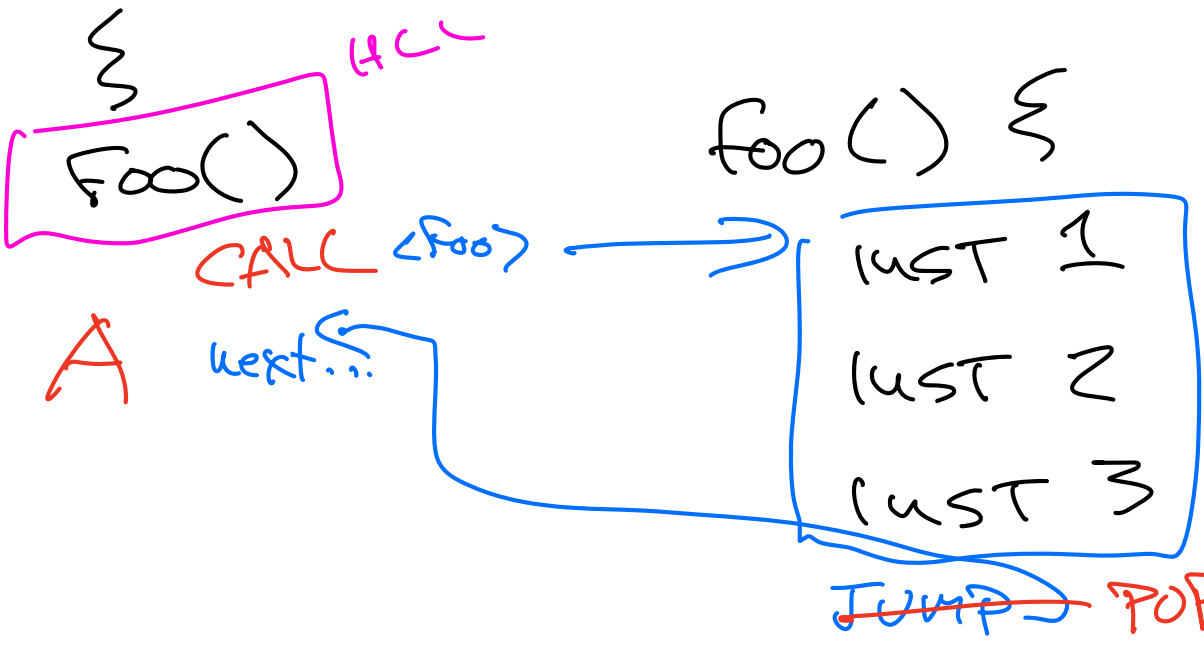
to <addr>

stack, CALL, RET

stack.push(item)

item = stack.pop()





PUSH Rxxx : push Rx onto stack

SP = SP - 2

STORE Rx → *(SP)

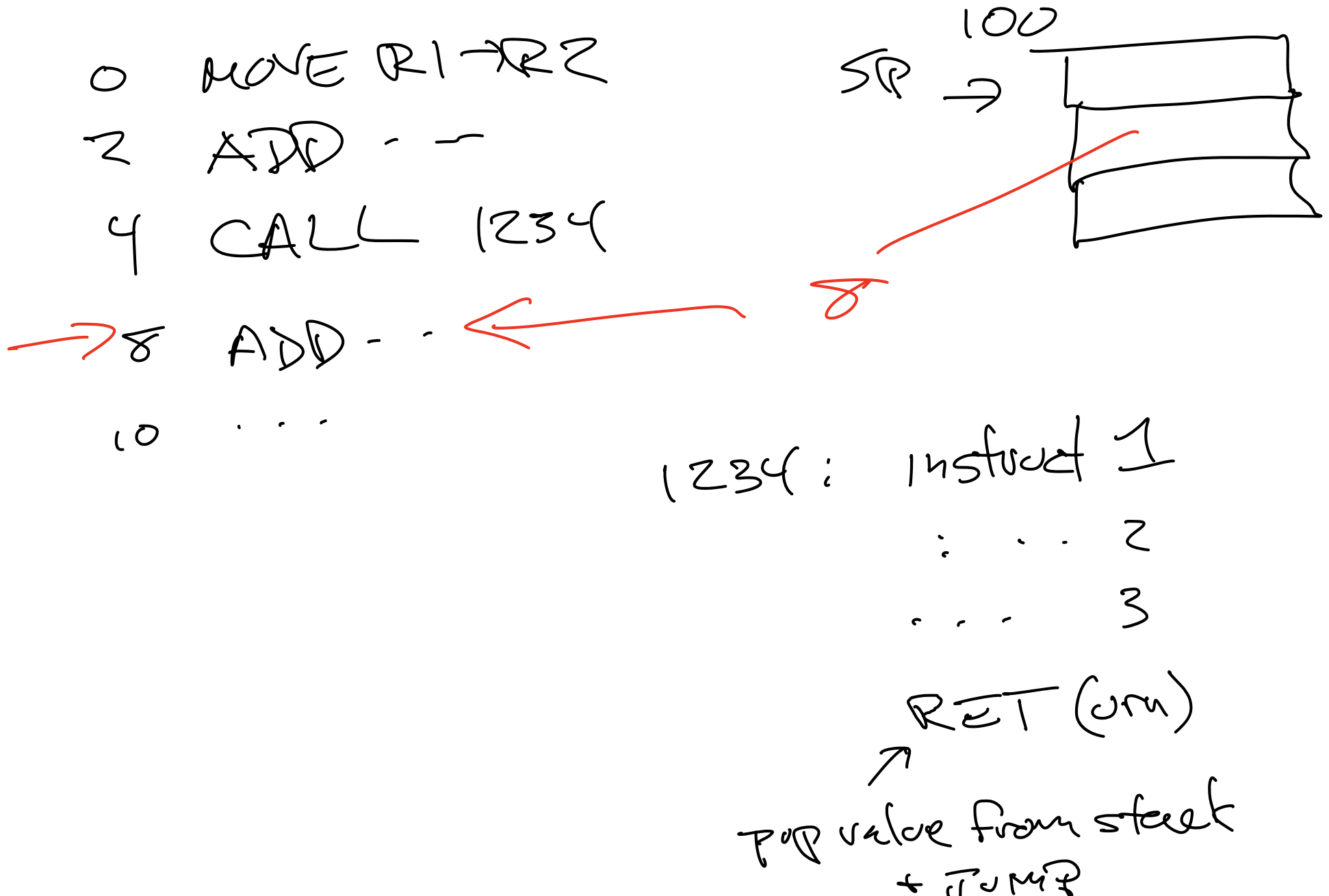
POP Rxxx

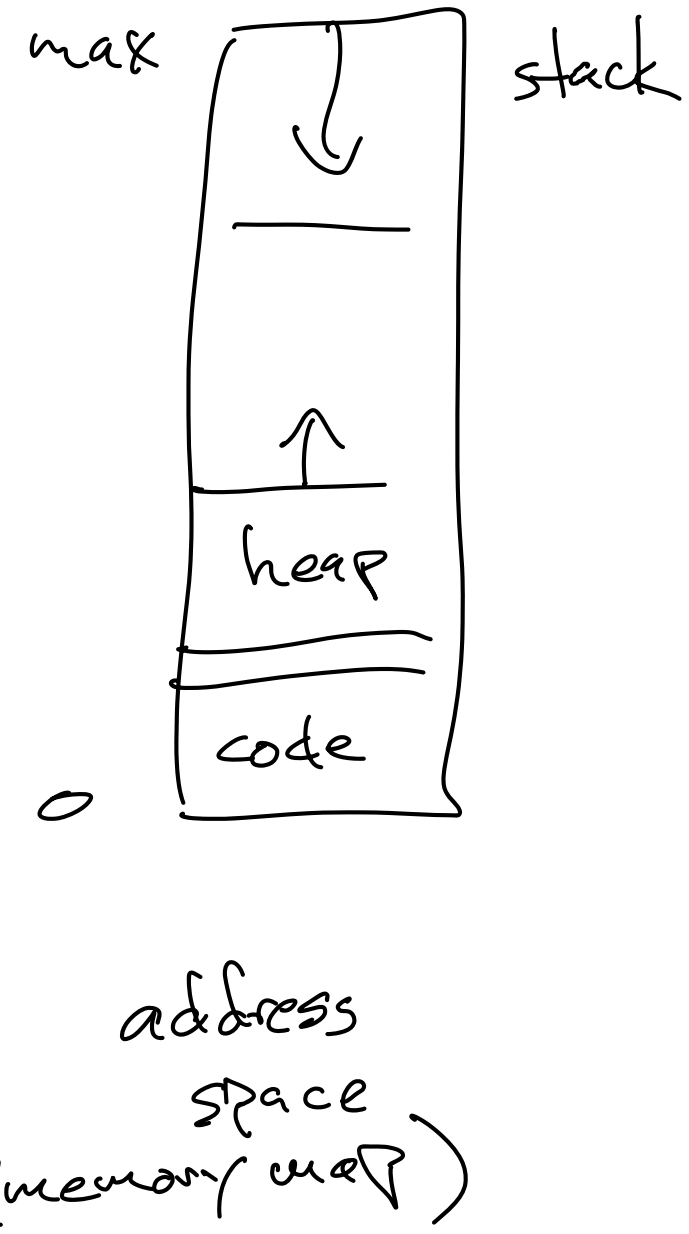
LOAD Rx ← *SP

SP = SP + 2

CALL <addr> : push addr of next instruction.

JUMP to <addr>


0 MOVE R1→R2
2 ADD - -
4 CALL 1234
8 ADD - -
10 . . .
100
SP →
1234: Instruct 1
. . . 2
. . . 3
RET (or)
pop value from steek
+ JUMP



address space (memory map)

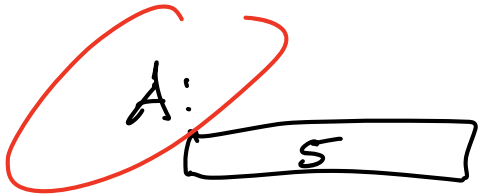
C Programming language

```
return-type functionname (type arg1, type arg2)
{
    type var1;
    type var2 = val;
    while,                    for loops
```

C variables

name
location
value

int a=5;



int *p = &a;

&a ← address of variable a

pointer:
address of a value

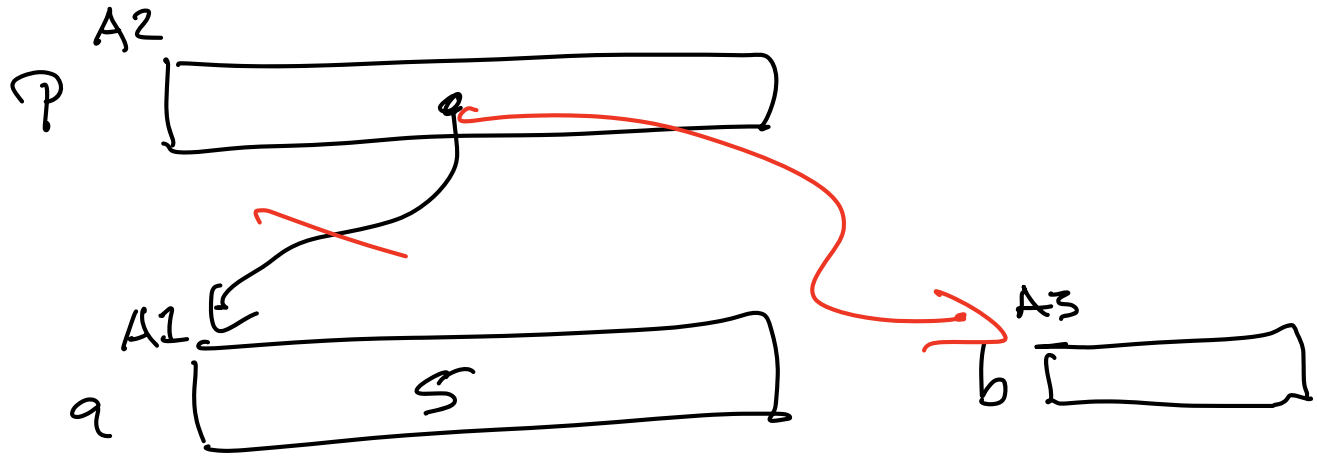
dereference:

b = *p

b == a (5)

*p = 7 → a == 7

```
int a=5;
int *p = &a;
```




```
int **q = &p
```

```
p = &b;
```

```
int c = *p
```

```
struct x *ptr_x;
```