Loops in Assembly Language

Computer Science 111
Boston University

Vahid Azadeh-Ranjbar, Ph.D.

based in part on notes from the CS-for-All curriculum developed at Harvey Mudd College

Jumps in Hmmm

instruction	what it does	example
jeqz rX L	jumps to line L if rX == 0	**jeqz r1 12**
jgtz rX L	jumps to line L if rX > 0	**jgtz r2 4**
jltz rX L	jumps to line L if rX < 0	**jltz r3 15**
jnez rX L	jumps to line L if rX != 0	**jnez r1 7**
jumpn L	jumps to line L	**jumpn 6**
jumpr rX	jumps to line # stored in rX	**jumpr r2**

(more on this later)

Notation:

- rX is any register name (**r1-r15**)
- L is the line number of an instruction

Instruction	Description	Aliases
System instructions		
halt	Stop!	
read rX	Place user input in register rX	
write rX	Print contents of register rX	
nop	Do nothing	
Setting register data		
setn rX N	Set register rX equal to the integer N (-128 to +127)	
addn rX N	Add integer N (-128 to 127) to register rX	
copy rX rY	Set rX = rY	mov
Arithmetic		
add rX rY rZ	Set rX = rY + rZ	
sub rX rY rZ	Set rX = rY - rZ	
neg rX rY	Set rX = -rY	
mul rX rY rZ	Set rX = rY * rZ	
div rX rY rZ	Set rX = rY / rZ (integer division; no remainder)	
mod rX rY rZ	Set rX = rY % rZ (returns the remainder of integer division)	
Jumps!		
jumpn N	Set program counter to address N	
jumpr rX	Set program counter to address in rX	jump
jeqzn rX N	If rX == 0, then jump to line N	jeqz
jnezn rX N	If rX != 0, then jump to line N	jnez
jgtzn rX N	If rX > 0, then jump to line N	jgtz
jltzn rX N	If rX < 0, then jump to line N	jltz
calln rX N	Copy the next address into rX and then jump to mem. addr. N	call
Interacting with memory (RAM)		
loadn rX N	Load register rX with the contents of memory address N	
storen rX N	Store contents of register rX into memory address N	
loadr rX rY	Load register rX with data from the address location held in reg. rY	loadi, load
storer rX rY	Store contents of register rX into memory address held in reg. rY	storei, store

Hmmm

the complete reference

www.cs.hmc.edu/~cs5grad/cs5/hmmm/documentation/documentation.html

What Does This Program Output?

Screen

-6 (input)

A. 6

B. −6

C. −7

D. 42

E. none of these

RAM

random access memory

0	`read r1`
1	`jgtz r1 7`
2	`setn r2 -1`
3	`mul r1 r1 r2`
4	`nop`
5	`nop`
6	`nop`
7	`write r1`
8	`halt`

space for future expansion!

What Does This Program Output?

Screen

-6 (input)

A. 6
B. −6
C. −7
D. 42
E. none of these

RAM

random access memory

0	`read r1`
1	`jgtz r1 7`
2	`setn r2 -1`
3	`mul r1 r1 r2`
4	`nop`
5	`nop`
6	`nop`
7	`write r1`
8	`halt`

space for future expansion!

What Does This Program Output?

Screen

-6 (input)
6 (output)

CPU ⟷ RAM

central processing unit

random access memory

r1	6
r2	-1

0	`read r1`
1	`jgtz r1 7`
2	`setn r2 -1`
3	`mul r1 r1 r2`
4	`nop`
5	`nop`
6	`nop`
7	`write r1`
8	`halt`

- What Python / mathematical function is this? `abs(x)`

What If We Enter a Positive Value?

Screen

8 (input)

CPU ⟷ RAM

central processing unit

random access memory

r1	
r2	

0	`read r1`
1	`jgtz r1 7`
2	`setn r2 -1`
3	`mul r1 r1 r2`
4	`nop`
5	`nop`
6	`nop`
7	`write r1`
8	`halt`

What If We Enter a Positive Value?

Screen

8 (input)

CPU ⟷ RAM

central processing unit

random access memory

r1	8
r2	

0	read r1
1	jgtz r1 7
2	setn r2 -1
3	mul r1 r1 r2
4	nop
5	nop
6	nop
7	write r1
8	halt

equivalent to:

```
if r1 > 0:
    jump to line 7
```

What If We Enter a Positive Value?

Screen

8 (input)

8 (output)

CPU ⟷ RAM

central processing unit

random access memory

r1	8
r2	

0	read r1
1	jgtz r1 7
2	setn r2 -1
3	mul r1 r1 r2
4	nop
5	nop
6	nop
7	write r1
8	halt

For the Inputs At Right, Which Lines Execute After the nop?

Screen

20 (1st input)
7 (2nd input)

0	`read r1`
1	`read r2`
2	`sub r3 r1 r2`
3	`nop`
4	`jgtz r3 7`
5	`write r1`
6	`jumpn 8`
7	`write r2`
8	`halt`

A. `4,5,6,8`
B. `4,7,8`
C. `4,5,6,7,8`
D. `4,6,8`
E. none of these

- What if we swap the two inputs?
- What Python function is this?
- How could you change <u>only line 3</u> so that, if the two inputs are equal, the program will ask for new inputs?

For the Inputs At Right, Which Lines Execute After the nop?

Screen
20 (1st input)
7 (2nd input)

0	`read r1`
1	`read r2`
2	`sub r3 r1 r2`
3	`nop`
4	`jgtz r3 7`
5	`write r1`
6	`jumpn 8`
7	`write r2`
8	`halt`

A. 4,5,6,8
B. **4,7,8**
C. 4,5,6,7,8
D. 4,6,8
E. none of these

- What if we swap the two inputs?
- What Python function is this?
- How could you change <u>only line 3</u> so that, if the two inputs are equal, the program will ask for new inputs?

For the Inputs At Right, Which Lines Execute After the nop?

Screen
20 (1st input)
7 (2nd input)

r1	20
r2	7
r3	13

0	`read r1`
1	`read r2`
2	`sub r3 r1 r2`
3	`nop`
4	`jgtz r3 7`
5	`write r1`
6	`jumpn 8`
7	`write r2`
8	`halt`

line 4 means:

```
if r3 > 0:
    jump to line 7
```

taken together,
lines 2 and 4 are equivalent to:

```
if r1 > r2:
    jump to line 7
```

For the Inputs At Right, Which Lines Execute After the nop?

Screen

20 (1st input)
7 (2nd input)

r1	20
r2	7
r3	13

0	read r1
1	read r2
2	sub r3 r1 r2
3	nop
4	jgtz r3 7
5	write r1
6	jumpn 8
7	write r2
8	halt

For the Inputs At Right, Which Lines Execute After the nop?

Screen

20 (1st input)
7 (2nd input)
7 (output)

r1	20
r2	7
r3	13

0	read r1
1	read r2
2	sub r3 r1 r2
3	nop
4	jgtz r3 7
5	write r1
6	jumpn 8
7	write r2
8	halt

For the Inputs At Right, Which Lines Execute After the nop?

Screen
7 (1st input)
20 (2nd input)

r1	7
r2	20
r3	-13

0	read r1
1	read r2
2	sub r3 r1 r2
3	nop
4	jgtz r3 7
5	write r1
6	jumpn 8
7	write r2
8	halt

- What if we swap the two inputs?

```
if r3 > 0:
    jump to line 7
```

For the Inputs At Right, Which Lines Execute After the nop?

Screen
7 (1st input)
20 (2nd input)
7 (output)

r1	7
r2	20
r3	-13

0	read r1
1	read r2
2	sub r3 r1 r2
3	nop
4	jgtz r3 7
5	write r1
6	jumpn 8
7	write r2
8	halt

- What if we swap the two inputs?

For the Inputs At Right, Which Lines Execute After the nop?

Screen
7 (1st input)
20 (2nd input)
7 (output)

r1	7
r2	20
r3	-13

0	read r1
1	read r2
2	sub r3 r1 r2
3	nop
4	jgtz r3 7
5	write r1
6	jumpn 8
7	write r2
8	halt

- What if we swap the two inputs?

For the Inputs At Right, Which Lines Execute After the nop?

Screen
7 (1st input)
20 (2nd input)
7 (output)

r1	7
r2	20
r3	-13

0	read r1
1	read r2
2	sub r3 r1 r2
3	nop
4	jgtz r3 7
5	write r1
6	jumpn 8
7	write r2
8	halt

- What if we swap the two inputs?
- What Python function is this? `min()`
- How could you change <u>only line 3</u> so that, if the two inputs are equal, the program will ask for new inputs?

For the Inputs At Right, Which Lines Execute After the nop?

Screen

7	(1st input)
20	(2nd input)
7	(output)

r1	7
r2	20
r3	-13

0	read r1
1	read r2
2	sub r3 r1 r2
3	jeqz r3 0
4	jgtz r3 7
5	write r1
6	jumpn 8
7	write r2
8	halt

- What if we swap the two inputs?
- What Python function is this? `min()`
- How could you change <u>only line 3</u> so that, if the two inputs are equal, the program will ask for new inputs?

Tracing What Happens For Equal Inputs

Screen

7	(1st input)
7	(2nd input)

r1	7
r2	
r3	

0	read r1
1	read r2
2	sub r3 r1 r2
3	jeqz r3 0
4	jgtz r3 7
5	write r1
6	jumpn 8
7	write r2
8	halt

Tracing What Happens For Equal Inputs

Screen

7 (1st input)
7 (2nd input)

r1	7
r2	7
r3	

0	`read r1`
1	`read r2`
2	`sub r3 r1 r2`
3	`jeqz r3 0`
4	`jgtz r3 7`
5	`write r1`
6	`jumpn 8`
7	`write r2`
8	`halt`

Tracing What Happens For Equal Inputs

Screen

7 (1st input)
7 (2nd input)

r1	7
r2	7
r3	0

0	`read r1`
1	`read r2`
2	`sub r3 r1 r2`
3	`jeqz r3 0`
4	`jgtz r3 7`
5	`write r1`
6	`jumpn 8`
7	`write r2`
8	`halt`

Tracing What Happens For Equal Inputs

Screen
7 (1st input)
7 (2nd input)

r1	7
r2	7
r3	0

0	`read r1`
1	`read r2`
2	`sub r3 r1 r2`
3	`jeqz r3 0`
4	`jgtz r3 7`
5	`write r1`
6	`jumpn 8`
7	`write r2`
8	`halt`

line 3 means:

```
if r3 == 0:
    jump to line 0
```

taken together,
lines 2 and 3 are equivalent to:

```
if r1 == r2:
    jump to line 0
```

Tracing What Happens For Equal Inputs

Screen
7 (1st input)
7 (2nd input)

r1	7
r2	7
r3	0

0	`read r1`
1	`read r2`
2	`sub r3 r1 r2`
3	`jeqz r3 0`
4	`jgtz r3 7`
5	`write r1`
6	`jumpn 8`
7	`write r2`
8	`halt`

Tracing What Happens For Equal Inputs

Screen
7 (1st input)
7 (2nd input)

r1	7
r2	7
r3	0

0	`read r1`
1	`read r2`
2	`sub r3 r1 r2`
3	`jeqz r3 0`
4	`jgtz r3 7`
5	`write r1`
6	`jumpn 8`
7	`write r2`
8	`halt`

- Lines 0-3 form a *loop.*
- When the inputs are equal, the program *loops back* and *repeats* those statements.
- It will continue to repeat them until the user enters non-equal inputs.

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

A. 2
B. 5
C. 6
D. 9
E. none of these

note:
`copy r3 r1`
is equivalent to:
`r3 = r1`

Use a table for the registers!
r1 r2 r3 r4

0	`read r1`
1	`read r2`
2	`copy r3 r1`
3	`sub r4 r2 r1`
4	`jeqz r4 12`
5	`nop`
6	`nop`
7	`nop`
8	`nop`
9	`addn r1 1`
10	`add r3 r3 r1`
11	`jumpn 3`
12	`write r3`
13	`halt`

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

A. 2
B. 5
C. 6
D. 9
E. none of these

note:
`copy r3 r1`
is equivalent to:
`r3 = r1`

- What does this program do in general?

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2			

- What does this program do in general?

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4		

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	2

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	2

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen

2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	2

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen

2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	2
3			

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	2
3		5	

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen
2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	2
3		5	

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen

2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	2
3		5	1

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen

2 (1st input)
4 (2nd input)

r1	r2	r3	r4
2	4	2	2
3		5	1
4		9	0

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen

2	(1st input)
4	(2nd input)
9	(output)

r1	r2	r3	r4
2	4	2	2
3		5	1
4		9	0

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- What does this program do in general?

For the Inputs At Right, What Is the Output of this Program?

Screen

2	(1st input)
4	(2nd input)
9	(output)

r1	r2	r3	r4
2	4	2	2
3		5	1
4		9	0

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- *What does this program do in general?*

For the Inputs At Right, What Is the Output of this Program?

Screen

2	(1st input)
4	(2nd input)
9	(output)

r1	r2	r3	r4
2	4	2	2
3		5	1
4		9	0

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

- *What does this program do in general?*
  compute the sum the integers from the 1st input to the 2nd input

Handling Invalid Inputs

Screen

4	(1st input)
2	(2nd input)

- What happens if the first input is larger than the second?
- Replace the nop instructions with ones that allow the program to still sum the integers in this case.
  - *hint:* you'll want to swap the values in `r1` and `r2`, but only if necessary
  - you may need to change another register, too

```
0  read r1
1  read r2
2  copy r3 r1
3  sub r4 r2 r1
4  jeqz r4 12
5  nop
6  nop
7  nop
8  nop
9  addn r1 1
10 add r3 r3 r1
11 jumpn 3
12 write r3
13 halt
```

Handling Invalid Inputs

Screen

4 (1st input)
2 (2nd input)

- What happens if the first input is larger than the second?
  the program will keep looping

- Replace the `nop` instructions with ones that allow the program to still sum the integers in this case.
  - *hint:* you'll want to swap the values in `r1` and `r2`, but only if necessary
  - you may need to change another register, too

0	`read r1`
1	`read r2`
2	`copy r3 r1`
3	`sub r4 r2 r1`
4	`jeqz r4 12`
5	`jgtz r4 9`
6	`copy r1 r2`
7	`copy r2 r3`
8	`copy r3 r1`
9	`addn r1 1`
10	`add r3 r3 r1`
11	`jumpn 3`
12	`write r3`
13	`halt`