Computer Organization

Assembly Pointers

Pointers in Assembly

- Since everything in memory is essentially a stack of address, we can very easily utilize the notion of pointers to dynamically adjust memory

- We store memory locations in our address register to acheive this functionality

Pointers

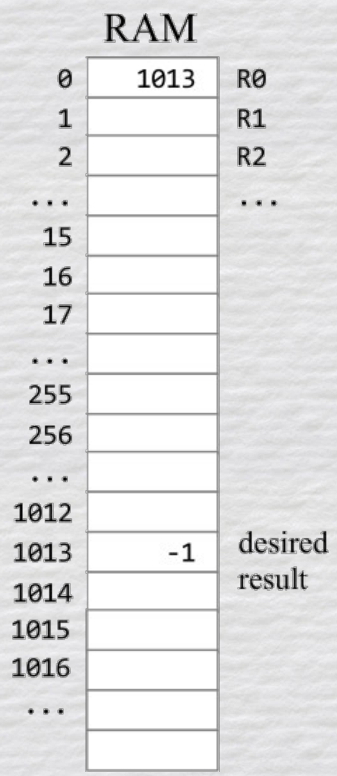
Example 1: Set the register at address *addr* to –1

Input: `R0`: Holds *addr*

```
// Sets RAM[R0] to -1
// Usage: Put some non-negative value in R0

@R0
A=M
M=-1
```

The key instruction:

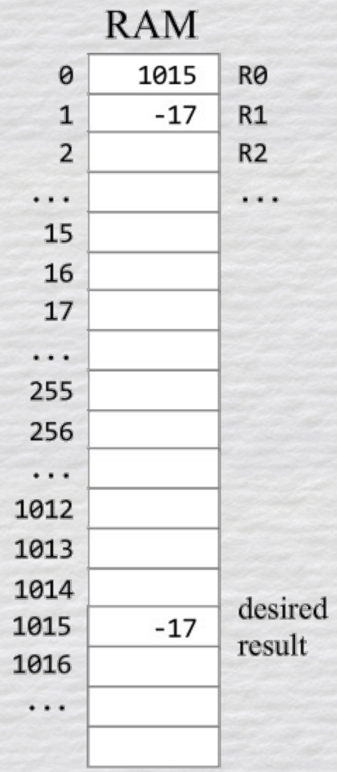
In the Hack machine language, pointer-based processing is realized by setting the address register to the address that we want to access, using the instruction `A=` ...



Pointers

Example 2:

```
// Sets RAM[R0] to R1
// Usage: Put some non-negative value in R0,
//             and some value in R1.

@R1

D=M

@R0

A=M

M=D
```

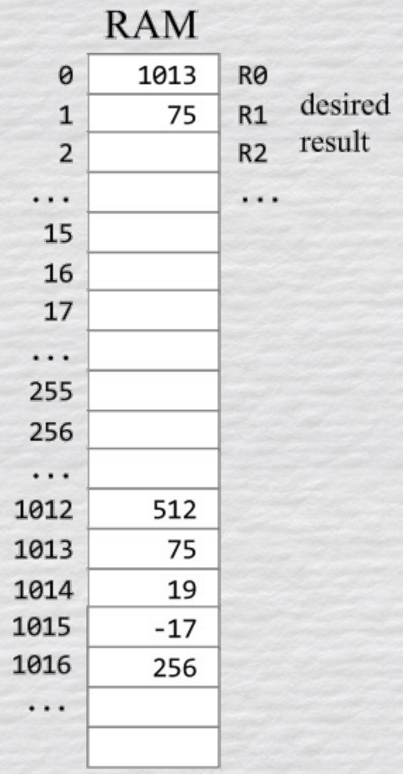


Pointers

Example 3: Get the value of the register at *addr*

Input: `R0`: Holds *addr*

```
// Sets R1 = RAM[R0]
// Usage: Put some non-negative value in R0
@R0
A=M
D=M
@R1
M=D
```



Pointers

Example 4: Set the first *n* entries of the memory block beginning in address *base* to `-1`

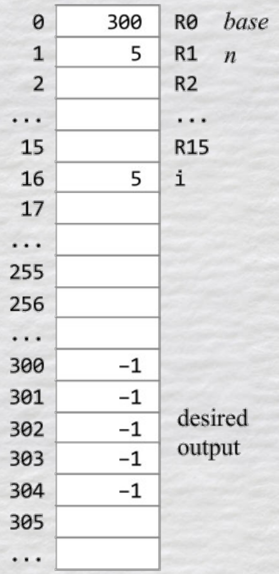
Inputs: `R0`: *base*
`R1`: *n*

Example: *base* = 300, *n* = 5

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
...
// RAM[R0 + i] = -1
@R0
D=M
@i
A=D+M
M=-1
...
```

The key operation

RAM



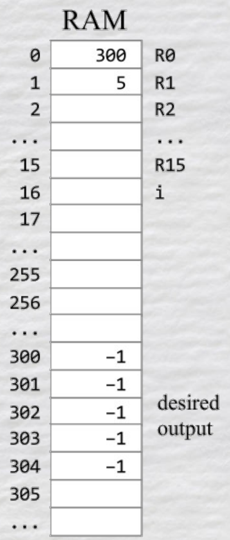
Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
```



0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	
302	-1	desired output
303	-1	
304	-1	
305		
...		

Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
    // i = 0
    @i
    M=0
```

RAM

0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	
302	-1	desired output
303	-1	
304	-1	
305		
...		

Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
    // i = 0
    @i
    M=0
(LOOP)
```

RAM

0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	
302	-1	desired output
303	-1	
304	-1	
305		
...		

Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
    // i = 0
    @i
    M=0
(LOOP)
    // if (i == R1) goto END
    @i
    D=M
    @R1
    D=D-M
    @END
    D;JEQ
```

RAM

0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	desired
302	-1	output
303	-1	
304	-1	
305		
...		

Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
    // i = 0
    @i
    M=0
(LOOP)
    // if (i == R1) goto END
    @i
    D=M
    @R1
    D=D-M
    @END
    D;JEQ
    // RAM[R0 + i] = -1
    @R0
    D=M
    @i
    A=D+M
    M=-1
```

RAM

0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	
302	-1	desired output
303	-1	
304	-1	
305		
...		

Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
    // i = 0
    @i
    M=0
(LOOP)
    // if (i == R1) goto END
    @i
    D=M
    @R1
    D=D-M
    @END
    D;JEQ
    // RAM[R0 + i] = -1
    @R0
    D=M
    @i
    A=D+M
    M=-1
    // i = i + 1
    @i
    M=M+1
```

RAM

0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	
302	-1	desired output
303	-1	
304	-1	
305		
...		

Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
    // i = 0
    @i
    M=0
(LOOP)
    // if (i == R1) goto END
    @i
    D=M
    @R1
    D=D-M
    @END
    D;JEQ
    // RAM[R0 + i] = -1
    @R0
    D=M
    @i
    A=D+M
    M=-1
    // i = i + 1
    @i
    M=M+1
    // goto LOOP
    @LOOP
    0;JMP
```

RAM

0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	
302	-1	desired output
303	-1	
304	-1	
305		
...		

Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
    // i = 0
    @i
    M=0
(LOOP)
    // if (i == R1) goto END
    @i
    D=M
    @R1
    D=D-M
    @END
    D;JEQ
    // RAM[R0 + i] = -1
    @R0
    D=M
    @i
    A=D+M
    M=-1
    // i = i + 1
    @i
    M=M+1
    // goto LOOP
    @LOOP
    0;JMP
(END)
```

RAM

0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	
302	-1	desired output
303	-1	
304	-1	
305		
...		

Pointer Procedure

Pseudocode

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
   i = 0
LOOP:
   if (i == R1) goto END
   RAM[R0 + i] = -1
   i = i + 1
   goto LOOP
END:
```

Assembly code

```
// Program: PointerDemo.asm
// Starting at the address stored in R0,
// sets the first R1 words to -1
    // i = 0
    @i
    M=0
(LOOP)
    // if (i == R1) goto END
    @i
    D=M
    @R1
    D=D-M
    @END
    D;JEQ
    // RAM[R0 + i] = -1
    @R0
    D=M
    @i
    A=D+M
    M=-1
    // i = i + 1
    @i
    M=M+1
    // goto LOOP
    @LOOP
    0;JMP
(END)
    @END
    0;JMP
```

RAM

0	300	R0
1	5	R1
2		R2
...		...
15		R15
16		i
17		
...		
255		
256		
...		
300	-1	
301	-1	
302	-1	desired output
303	-1	
304	-1	
305		
...		