We acknowledge and pay our respects to the Kaurna people,
the traditional custodians whose ancestral lands we gather on.

We acknowledge the deep feelings of attachment and relationship of the Kaurna people to country and we respect and value their past, present and ongoing connection to the land and cultural beliefs.

# Computer Systems

Lecture 12: Virtual Machine
Review and Exercise

# Memory segments and access commands

The VM abstraction includes 8 separate memory segments named:
static, this, local, argument, that, constant, pointer, temp

As far as VM programming commands go, all memory segments look and behave the same

To access a particular segment entry, use the following generic syntax:

Memory access VM commands:

- ❑ pop *memorySegment index*
- ❑ push *memorySegment index*

Where *memorySegment* is `static`, `this`, `local`, `argument`, `that`, `constant`, `pointer`, or `temp`

And *index* is a non-negative integer

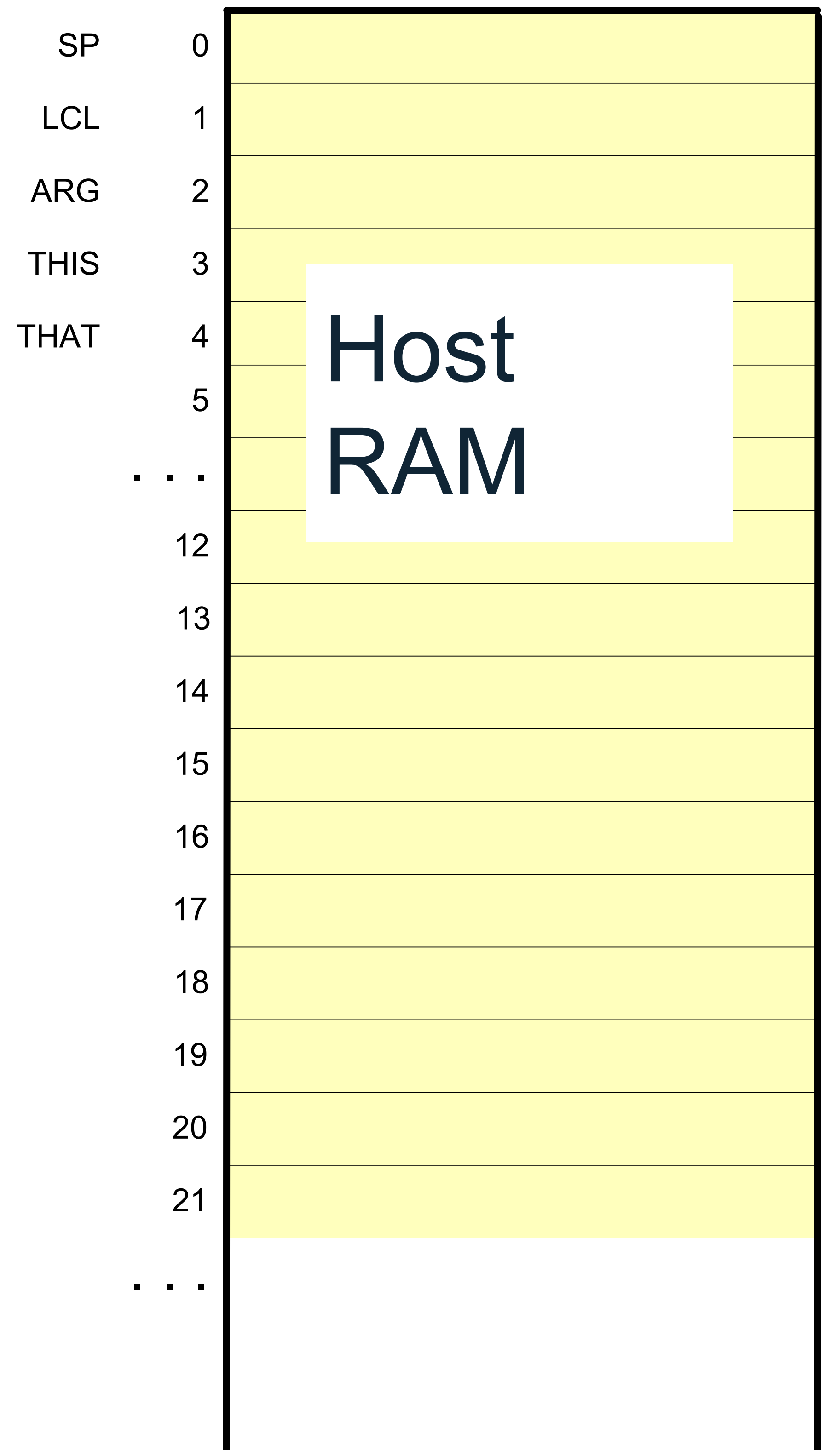
**Notes:**

(In all our code examples thus far, *memorySegment* was `static`)

The roles of the eight memory segments will become relevant when we talk about compiling

At the VM abstraction level, all memory segments are treated the same way.

# VM implementation on the Hack platform



The stack: a global data structure, used to save and restore the resources of all the VM functions up the calling hierarchy.

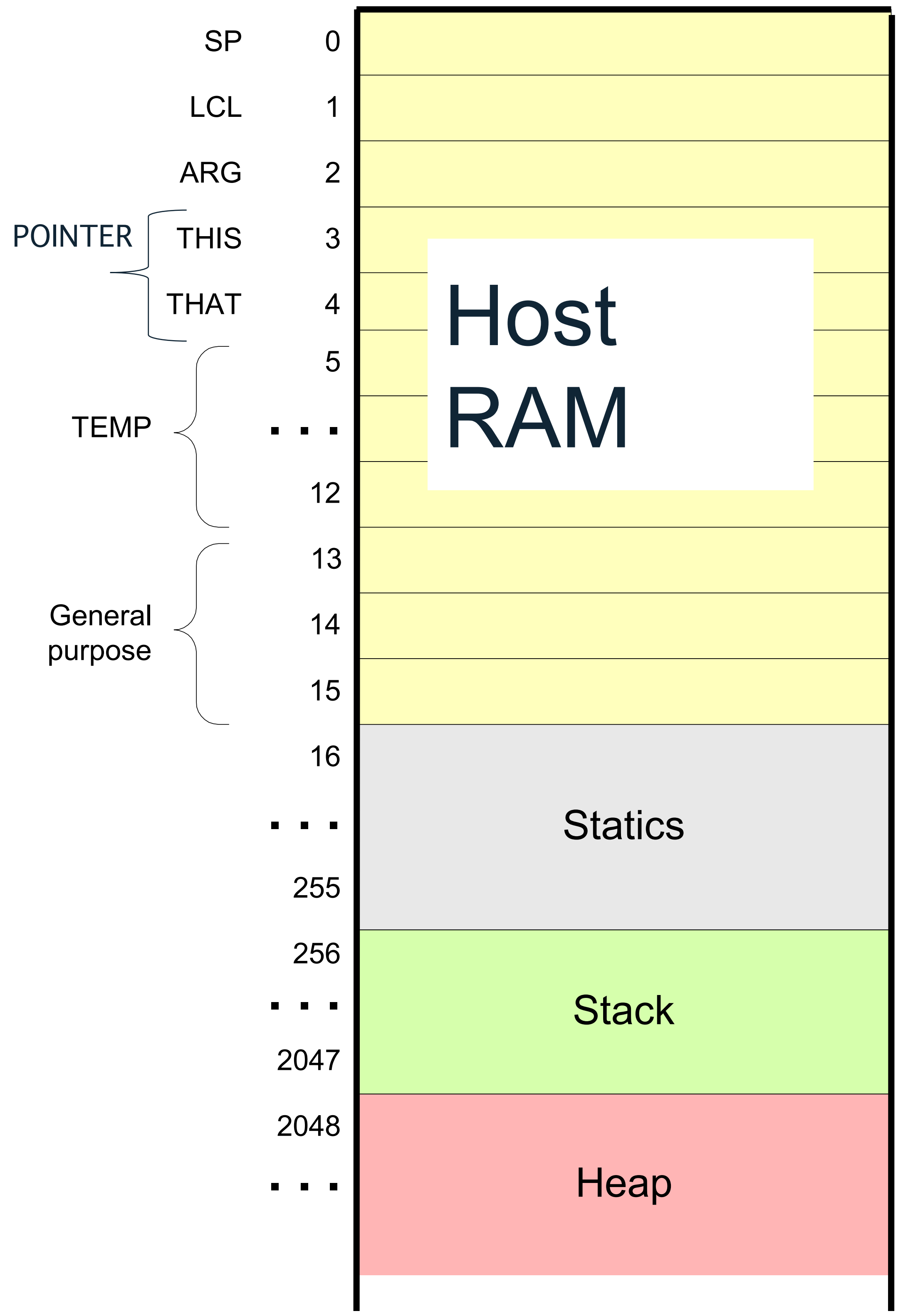
The tip of this stack is the working stack of the current function

static, constant, temp, pointer:
Global memory segments, all functions see the same four segments

local,argument,this,that:
these segments are local at the function level; each function sees its own, private copy of each one of these four segments

The challenge:
represent all these logical constructs on the same single physical address space -- the host RAM.

# VM implementation on the Hack platform



**Basic idea:** the mapping of the stack and the global segments on the RAM is easy (fixed); the mapping of the function-level segments is dynamic, using pointers

The stack: mapped on RAM[256 ... 2047]; The stack pointer is kept in RAM address SP

static: mapped on RAM[16 ... 255]; each segment reference static *i* appearing in a VM file named f is compiled to the assembly language symbol f.i (recall that the assembler further maps such symbols to the RAM, from address 16 onward)

local,argument,this,that: these method-level segments are mapped somewhere from address 256 onward, on the "stack" or the "heap". The base addresses of these segments are kept in RAM addresses LCL, ARG, THIS, and THAT. Access to the *i*-th entry of any of these segments is implemented by accessing RAM[segmentBase + *i*]

constant: a truly virtual segment: access to constant *i* is implemented by supplying the constant *i.*

pointer: RAM[3..4] to change THIS and THAT.

THE UNIVERSITY of ADELAIDE

## VM Translator Parsing

*You will implement this in assignment*

- **`push constant 1`**

```
@SP
AM=M+1
A=A-1
M=1
```

- **`pop static 7 (in a VM file named Bob.vm)`**

```
@SP
AM=M-1
D=M
@Bob.7
M=D
```

# VM Translator Parsing

*You will implement this in assignment*

- **push constant 5**

```
@5
D=A
@SP
AM=M+1
A=A-1
M=D
```

- **add**

```
@SP
AM=M-1
D=M
A=A-1
M=D+M
```

- **pop local 2**

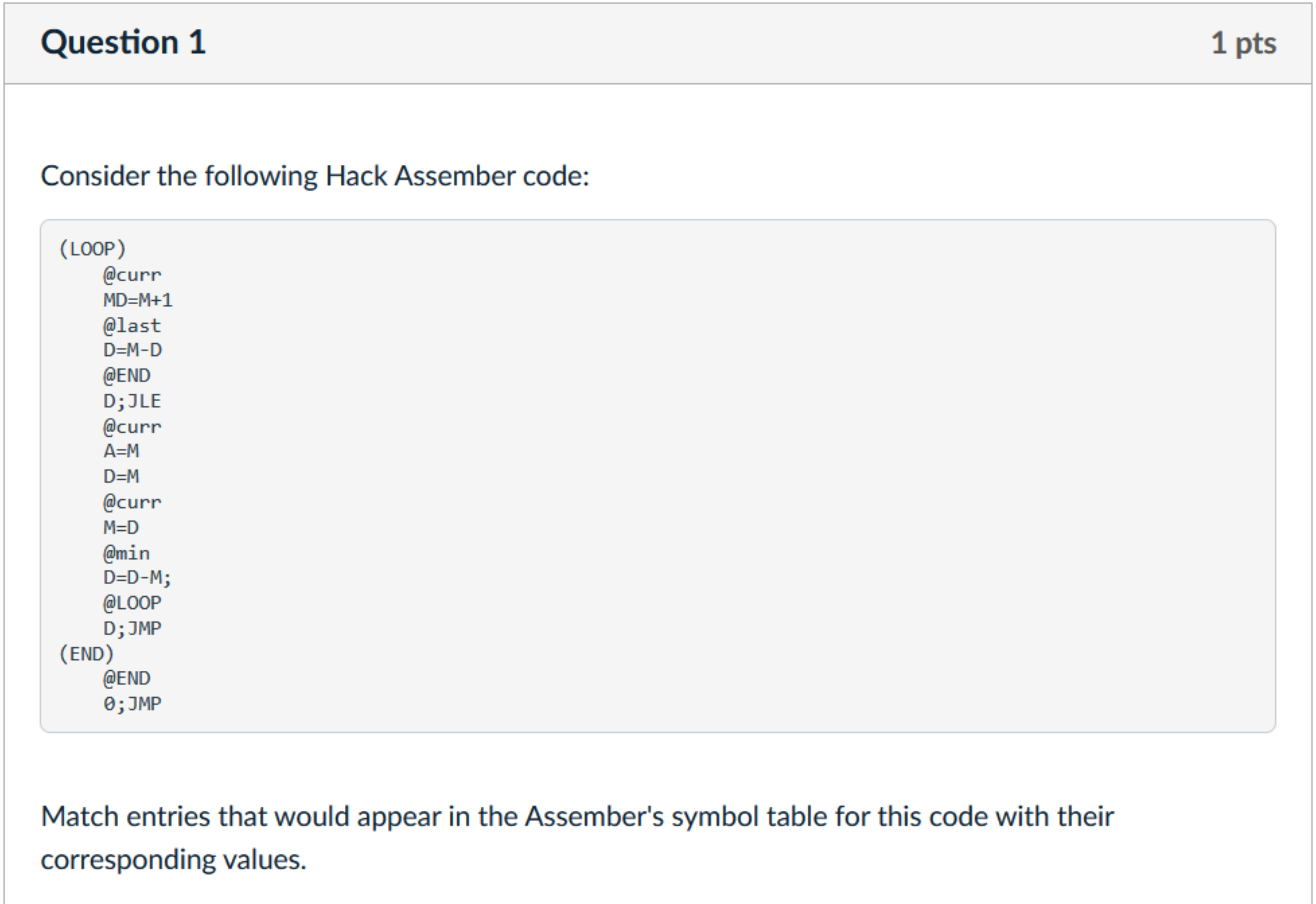
```
@SP
AM=M-1
D=M
@LOCAL
A=A+1
A=A+1
M=D
```

## VM Translator Parsing

*You will implement this in assignment*

- **eq**

```
@SP
AM=M-1
D=M
@SP
AM=M-1
D=M-D
@labelTrue
D;JEQ
D=0
@labelFalse
0;JMP
(labelTrue)
D=-1
(labelFalse)
@SP
A=M
M=D
@SP
M=M+1
```



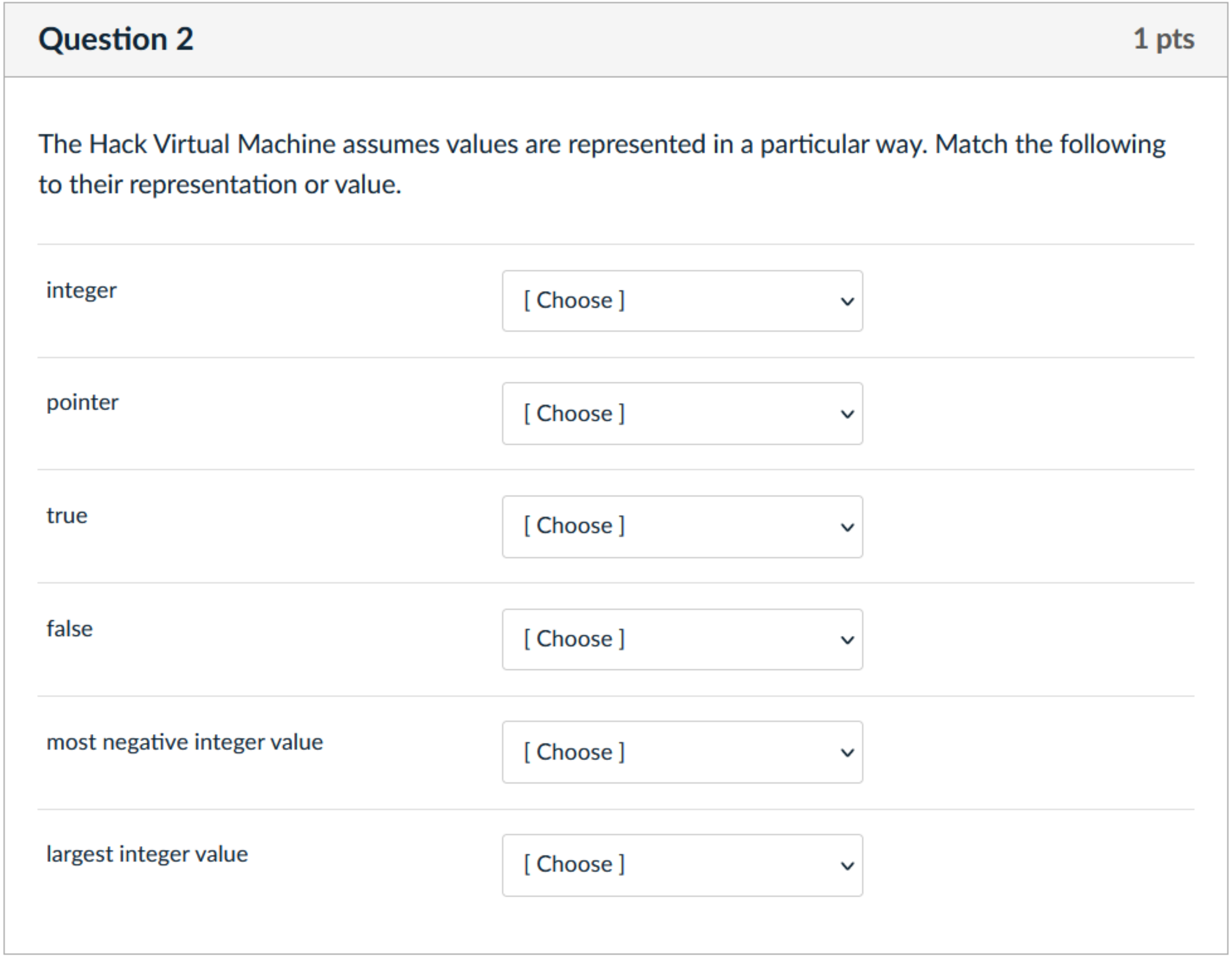
## Question 1

1 pts

Consider the following Hack Assember code:

```
(LOOP)
    @curr
    MD=M+1
    @last
    D=M-D
    @END
    D;JLE
    @curr
    A=M
    D=M
    @curr
    M=D
    @min
    D=D-M;
    @LOOP
    D;JMP
(END)
    @END
    0;JMP
```

Match entries that would appear in the Assember's symbol table for this code with their corresponding values.



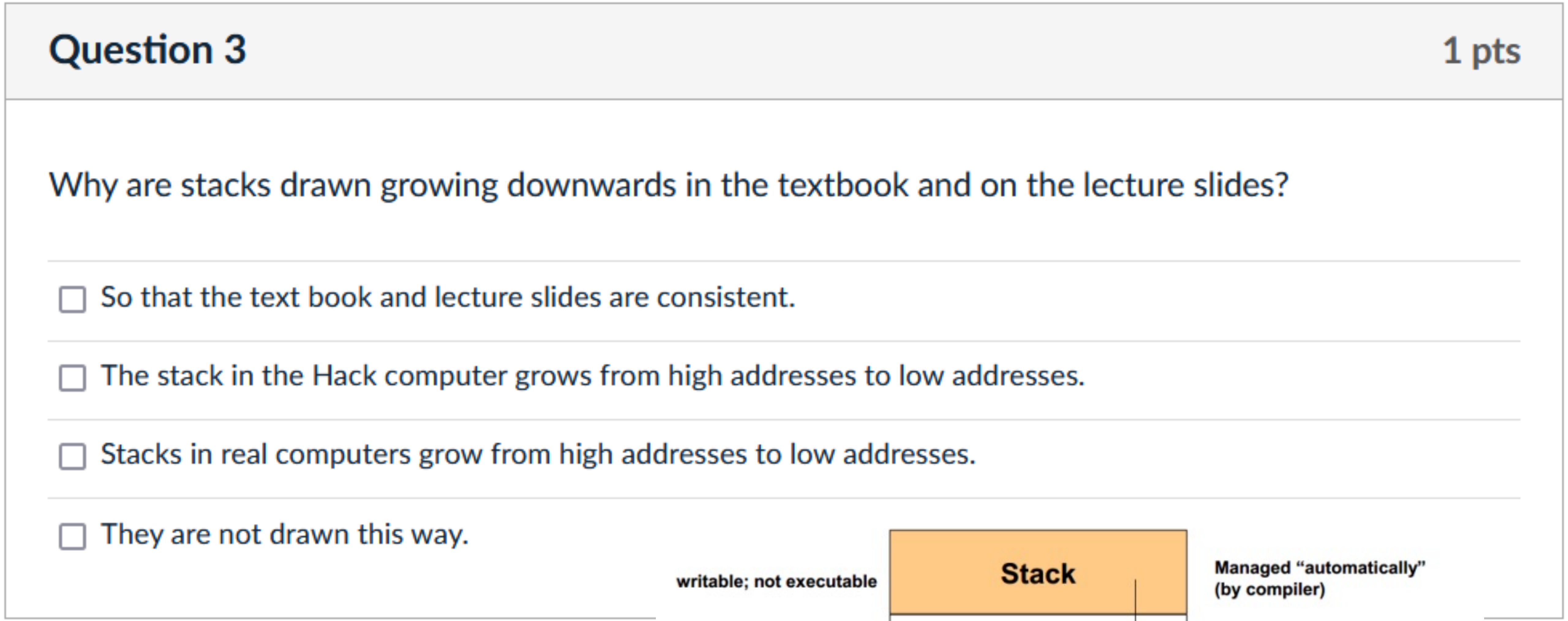
## Question 2

1 pts

The Hack Virtual Machine assumes values are represented in a particular way. Match the following to their representation or value.

| | |
|---|---|
| integer | [ Choose ] |
| pointer | [ Choose ] |
| true | [ Choose ] |
| false | [ Choose ] |
| most negative integer value | [ Choose ] |
| largest integer value | [ Choose ] |

THE UNIVERSITY of ADELAIDE

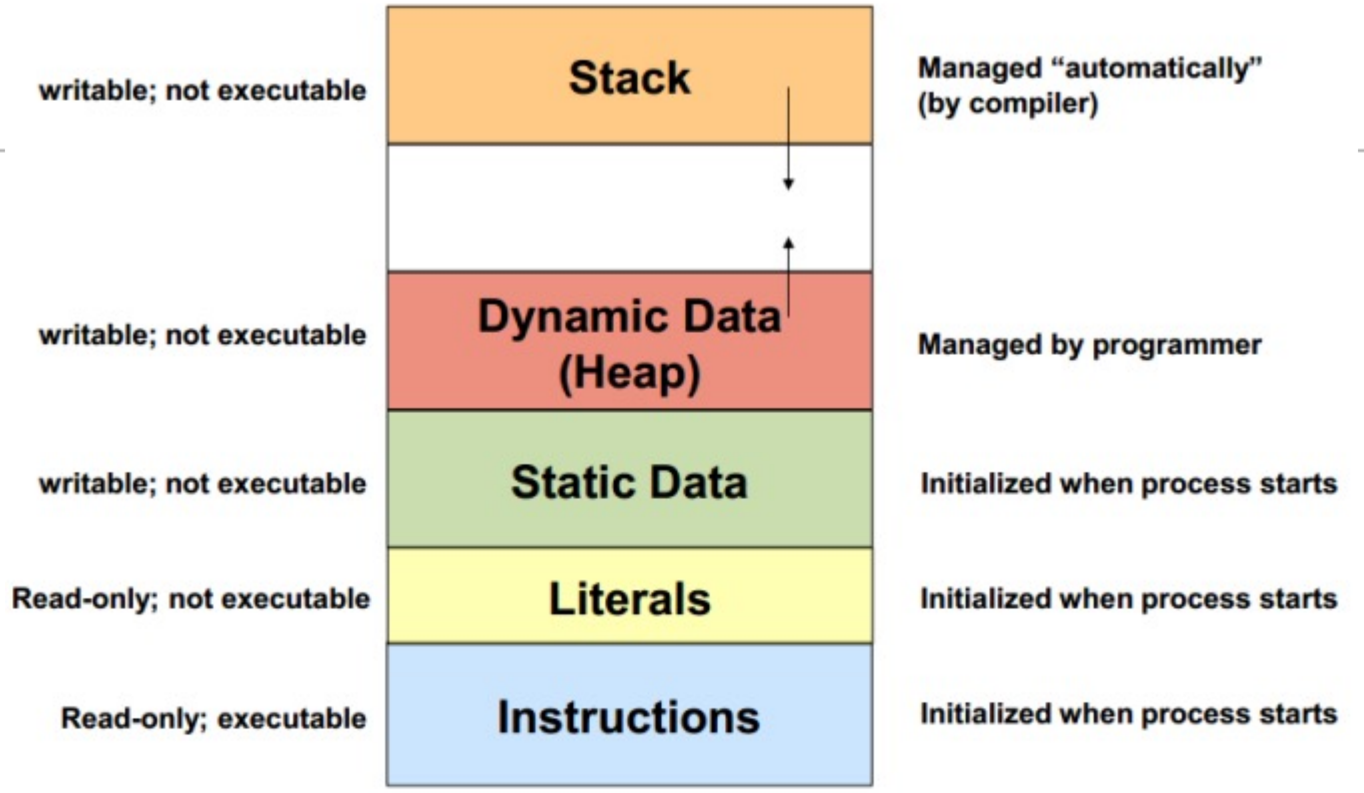


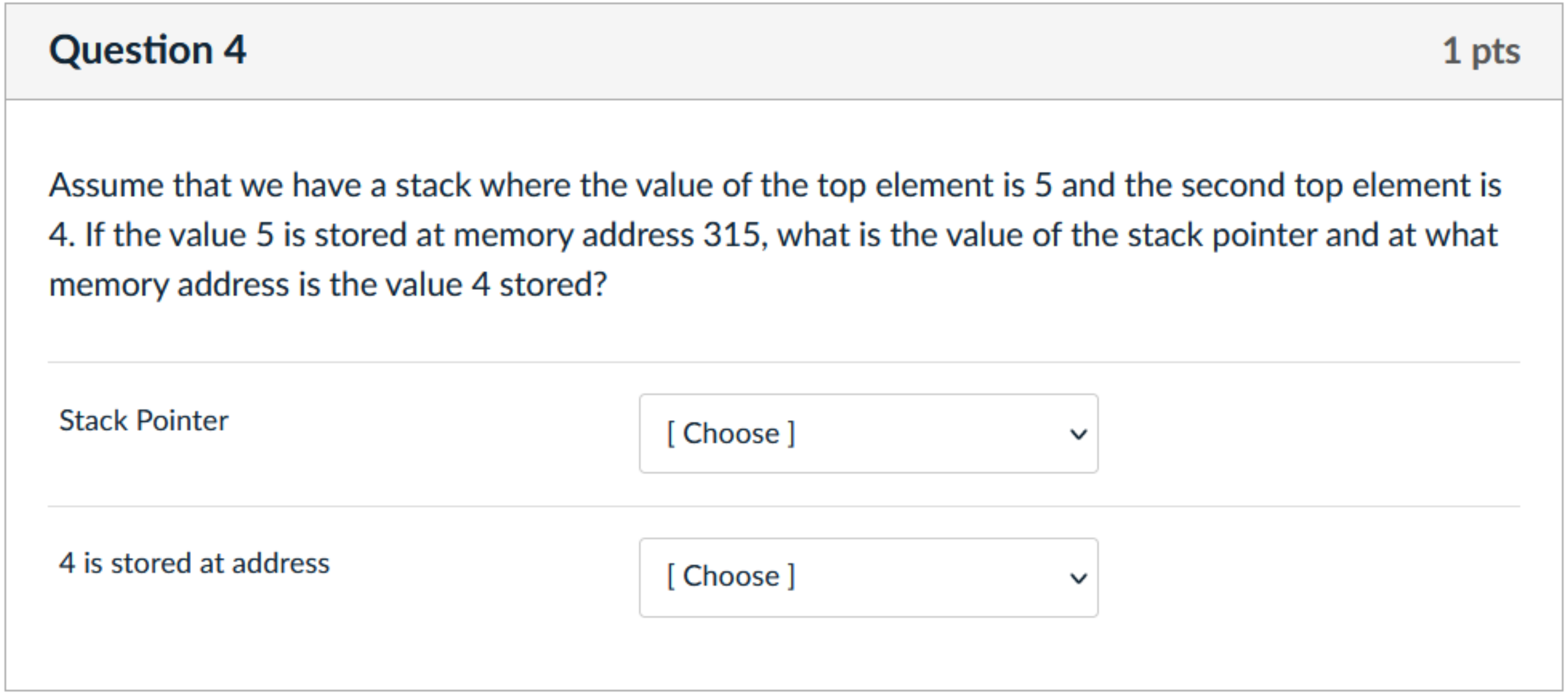
## Question 3

1 pts

Why are stacks drawn growing downwards in the textbook and on the lecture slides?

- ☐ So that the text book and lecture slides are consistent.
- ☐ The stack in the Hack computer grows from high addresses to low addresses.
- ☐ Stacks in real computers grow from high addresses to low addresses.
- ☐ They are not drawn this way.



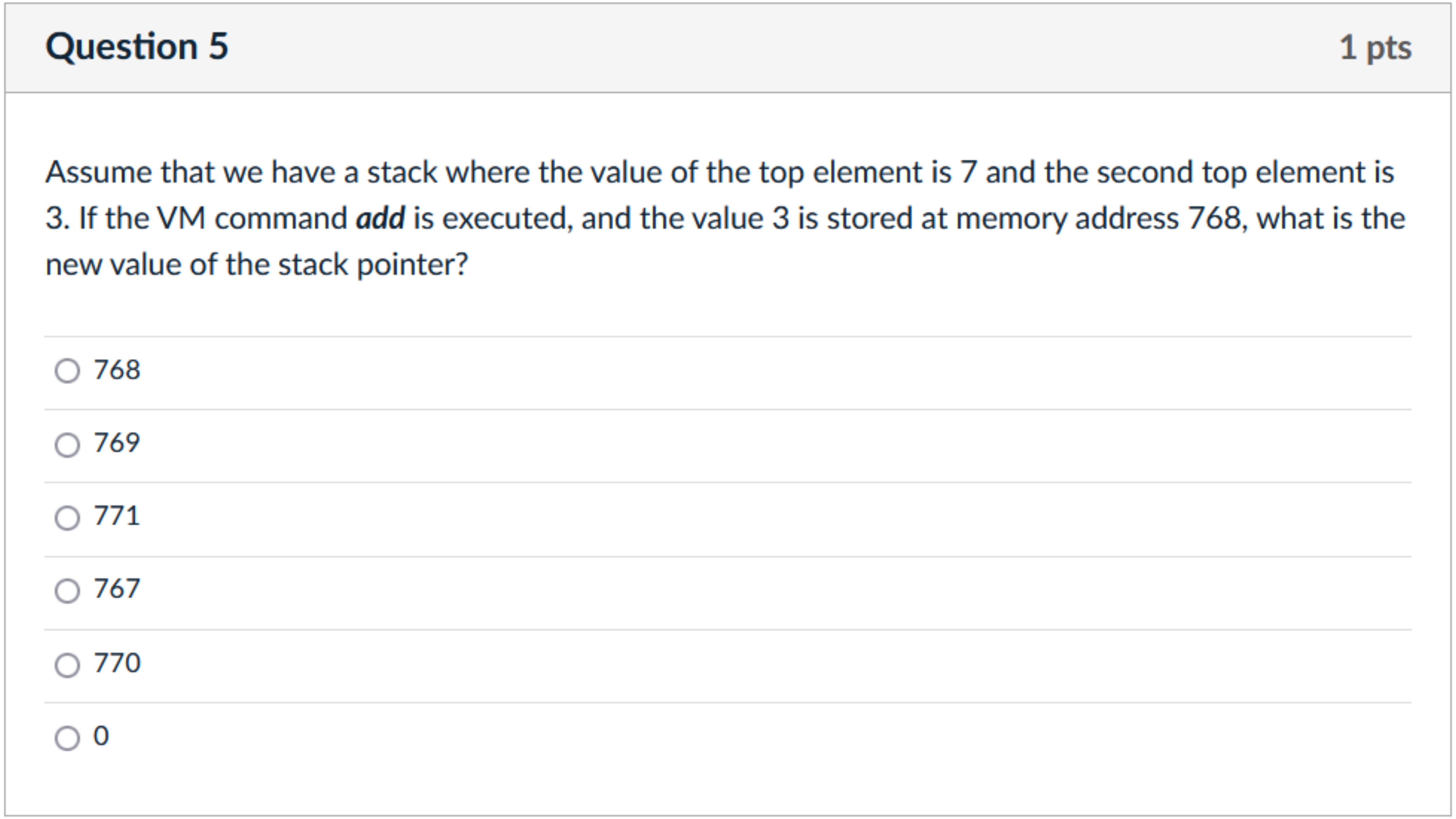


## Question 4

1 pts

Assume that we have a stack where the value of the top element is 5 and the second top element is 4. If the value 5 is stored at memory address 315, what is the value of the stack pointer and at what memory address is the value 4 stored?

| | |
|---|---|
| Stack Pointer | [ Choose ] |
| 4 is stored at address | [ Choose ] |



## Question 5

1 pts

Assume that we have a stack where the value of the top element is 7 and the second top element is 3. If the VM command ***add*** is executed, and the value 3 is stored at memory address 768, what is the new value of the stack pointer?

- ○ 768
- ○ 769
- ○ 771
- ○ 767
- ○ 770
- ○ 0

THE UNIVERSITY *of* ADELAIDE

## Exercise:

*You will see similar question in exam*

`VM command` **`not`** `in hack assembly (3 lines):`

## Exercise:

*You will see similar question in exam*

Write the VM commands to compute **((a+b)+c)**:
Suppose **a**, **b** and **c** are in the local segment at offsets **4**, **5** and **6**.

## Exercise:

*You will see similar question in exam*

For the same computation **((a+b)+c)**, draw the top of the stack Before and after each of the two **+** operation.
Suppose **a**, **b** and **c** are in the local segment at offsets **4**, **5** and **6**.

## This Week

- Review Chapters 6 & 7 of the Text Book (if you haven’t already)
- Assignment 4 due this Sunday.
- Assignment 5 Available next week.
- Review Chapter 8 of the Text Book before week 7.

THE UNIVERSITY of ADELAIDE