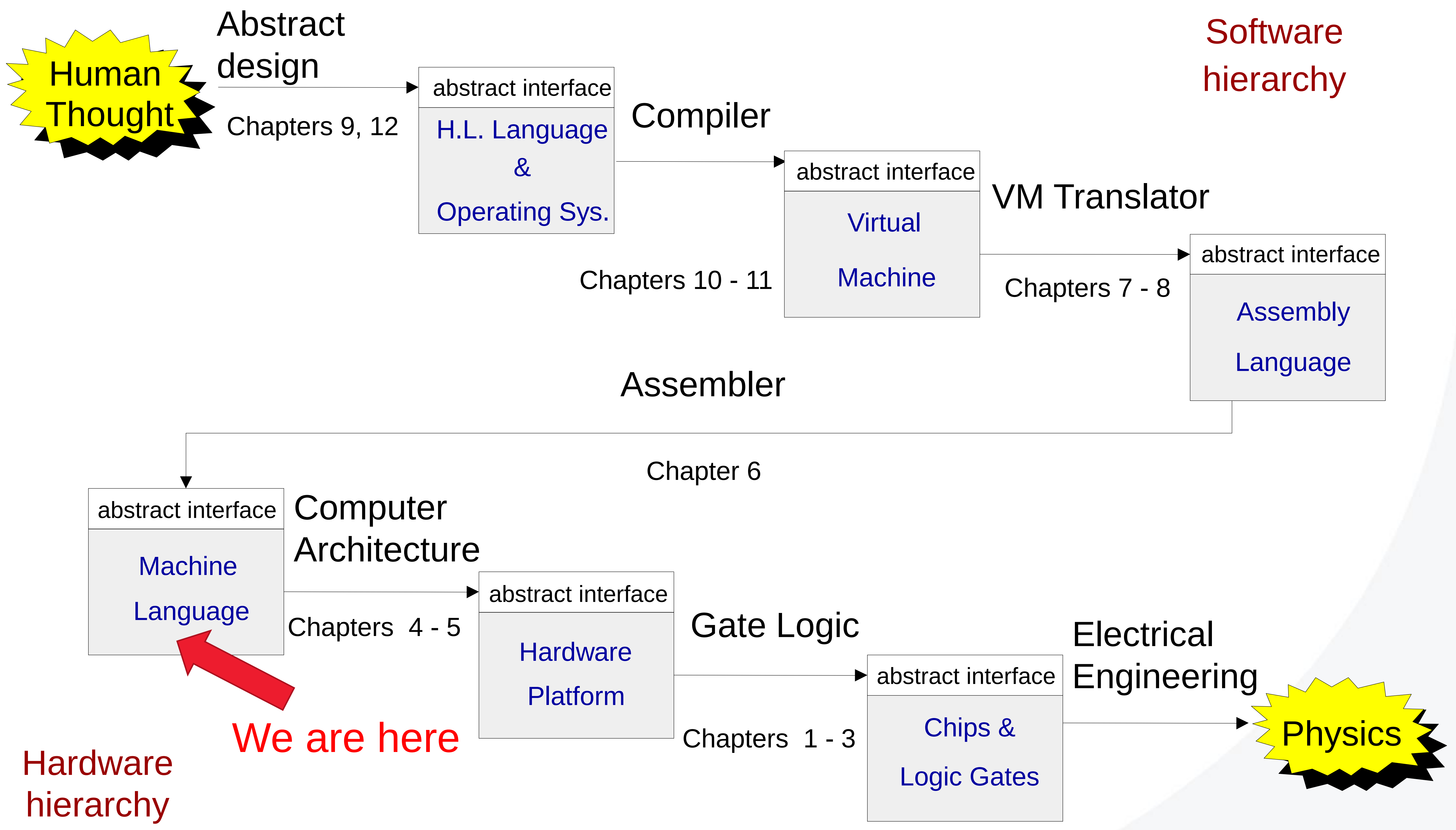
Computer Systems

Lecture 08: Machine Language

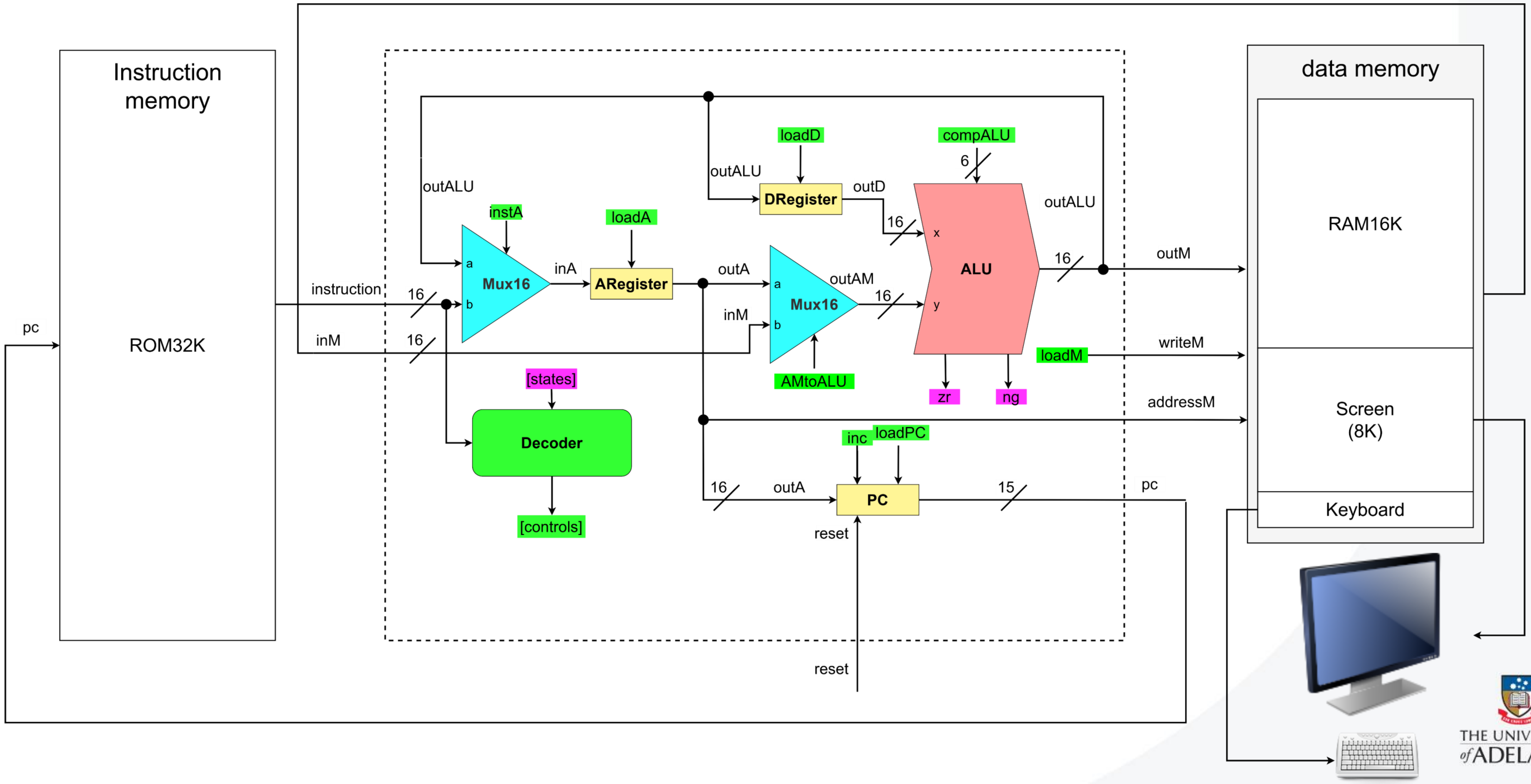
Review and Exercise

Our Journey



(Abstraction–implementation paradigm)

Review: The Hack Computer

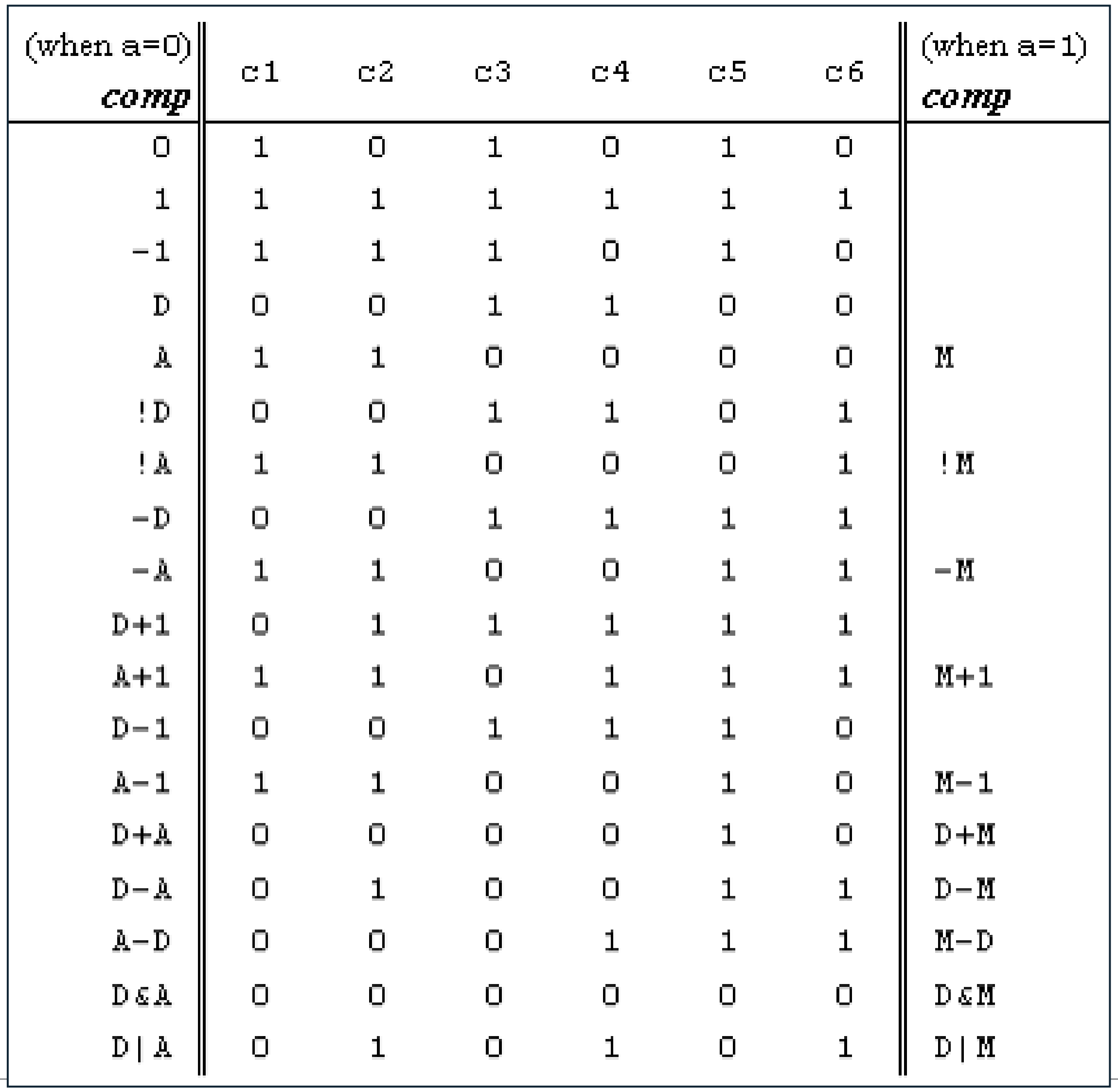


Review: The ***C***-instruction

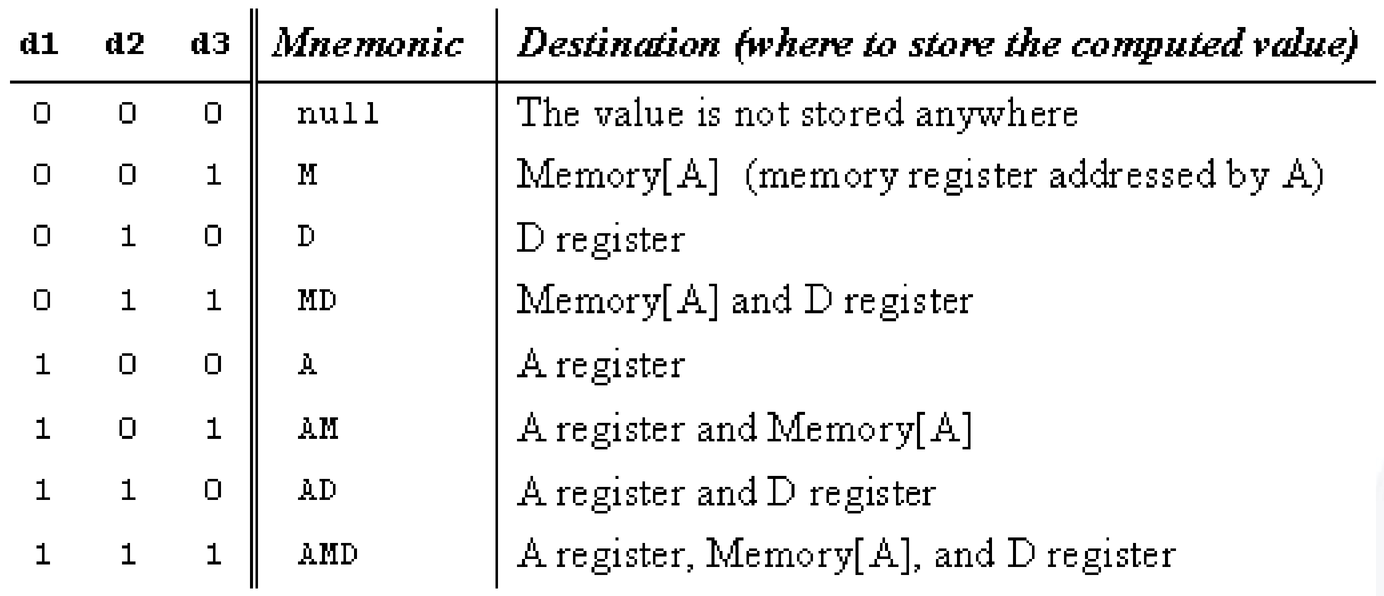
In C-instruction, we have *dest* field, why we still have A-instruction?

dest = comp; jump

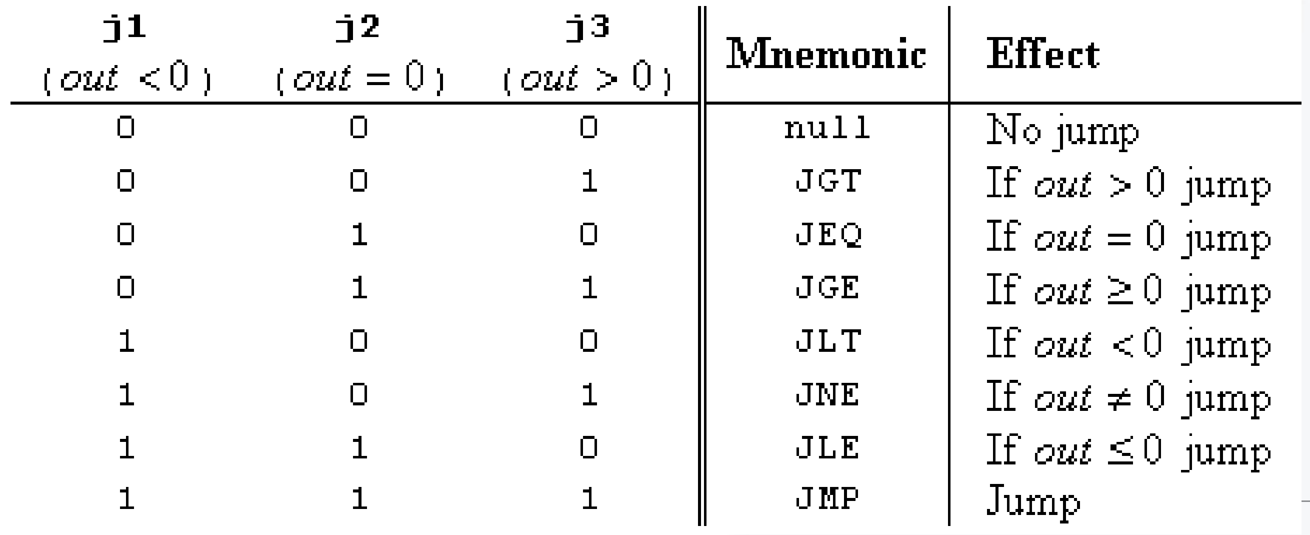
					comp						***dest***			***jump***		
binary:	**1**	**1**	**1**	a	c1	c2	c3	c4	c5	c6	d1	d2	d3	j1	j2	j3
	i15	**i14**	**i13**	**i12**	**i11**	**i10**	**i9**	**i8**	**i7**	**i6**	**i5**	**i4**	**i3**	**i2**	**i1**	**i0**



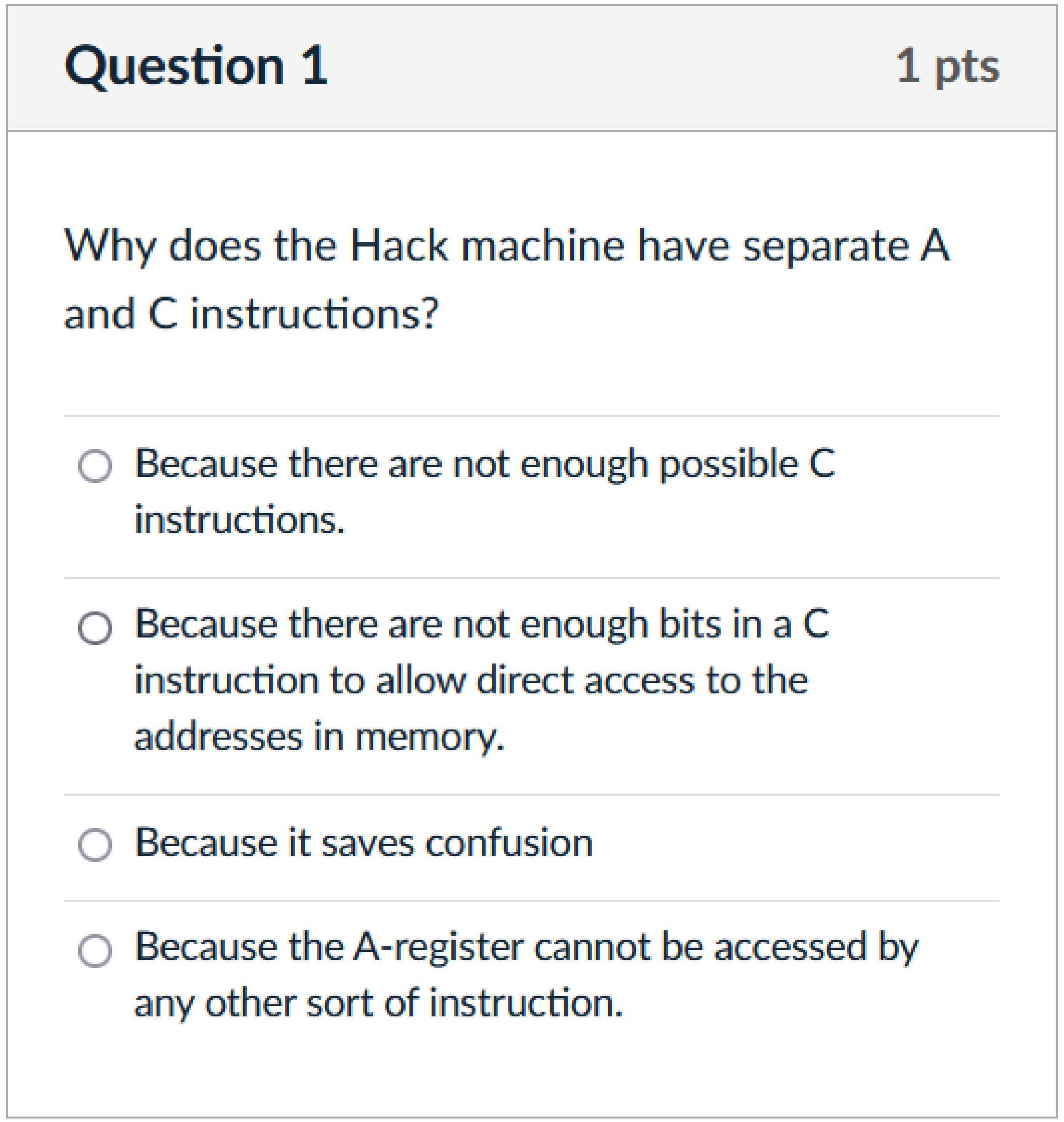
(when a=0) *comp*	c1	c2	c3	c4	c5	c6	(when a=1) *comp*
0	1	0	1	0	1	0	
1	1	1	1	1	1	1	
-1	1	1	1	0	1	0	
D	0	0	1	1	0	0	
A	1	1	0	0	0	0	M
!D	0	0	1	1	0	1	
!A	1	1	0	0	0	1	!M
-D	0	0	1	1	1	1	
-A	1	1	0	0	1	1	-M
D+1	0	1	1	1	1	1	
A+1	1	1	0	1	1	1	M+1
D-1	0	0	1	1	1	0	
A-1	1	1	0	0	1	0	M-1
D+A	0	0	0	0	1	0	D+M
D-A	0	1	0	0	1	1	D-M
A-D	0	0	0	1	1	1	M-D
D&A	0	0	0	0	0	0	D&M
D|A	0	1	0	1	0	1	D|M



d1	d2	d3	*Mnemonic*	*Destination (where to store the computed value)*
0	0	0	null	The value is not stored anywhere
0	0	1	M	Memory[A] (memory register addressed by A)
0	1	0	D	D register
0	1	1	MD	Memory[A] and D register
1	0	0	A	A register
1	0	1	AM	A register and Memory[A]
1	1	0	AD	A register and D register
1	1	1	AMD	A register, Memory[A], and D register



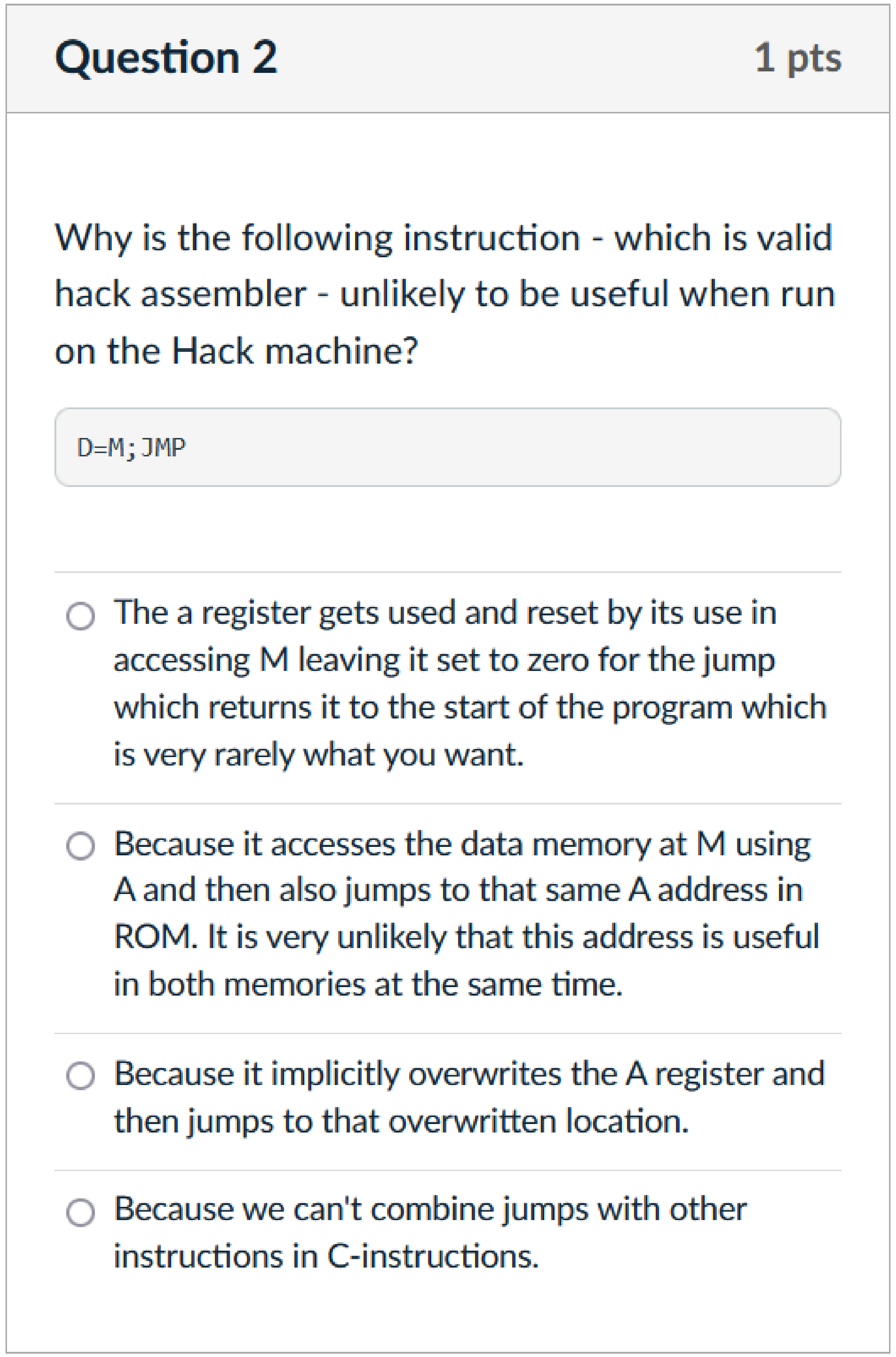
j1 ($out < 0$)	j2 ($out = 0$)	j3 ($out > 0$)	Mnemonic	Effect
0	0	0	null	No jump
0	0	1	JGT	If $out > 0$ jump
0	1	0	JEQ	If $out = 0$ jump
0	1	1	JGE	If $out \geq 0$ jump
1	0	0	JLT	If $out < 0$ jump
1	0	1	JNE	If $out \neq 0$ jump
1	1	0	JLE	If $out \leq 0$ jump
1	1	1	JMP	Jump



Question 1 1 pts

Why does the Hack machine have separate A and C instructions?

- ○ Because there are not enough possible C instructions.
- ○ Because there are not enough bits in a C instruction to allow direct access to the addresses in memory.
- ○ Because it saves confusion
- ○ Because the A-register cannot be accessed by any other sort of instruction.



Question 2

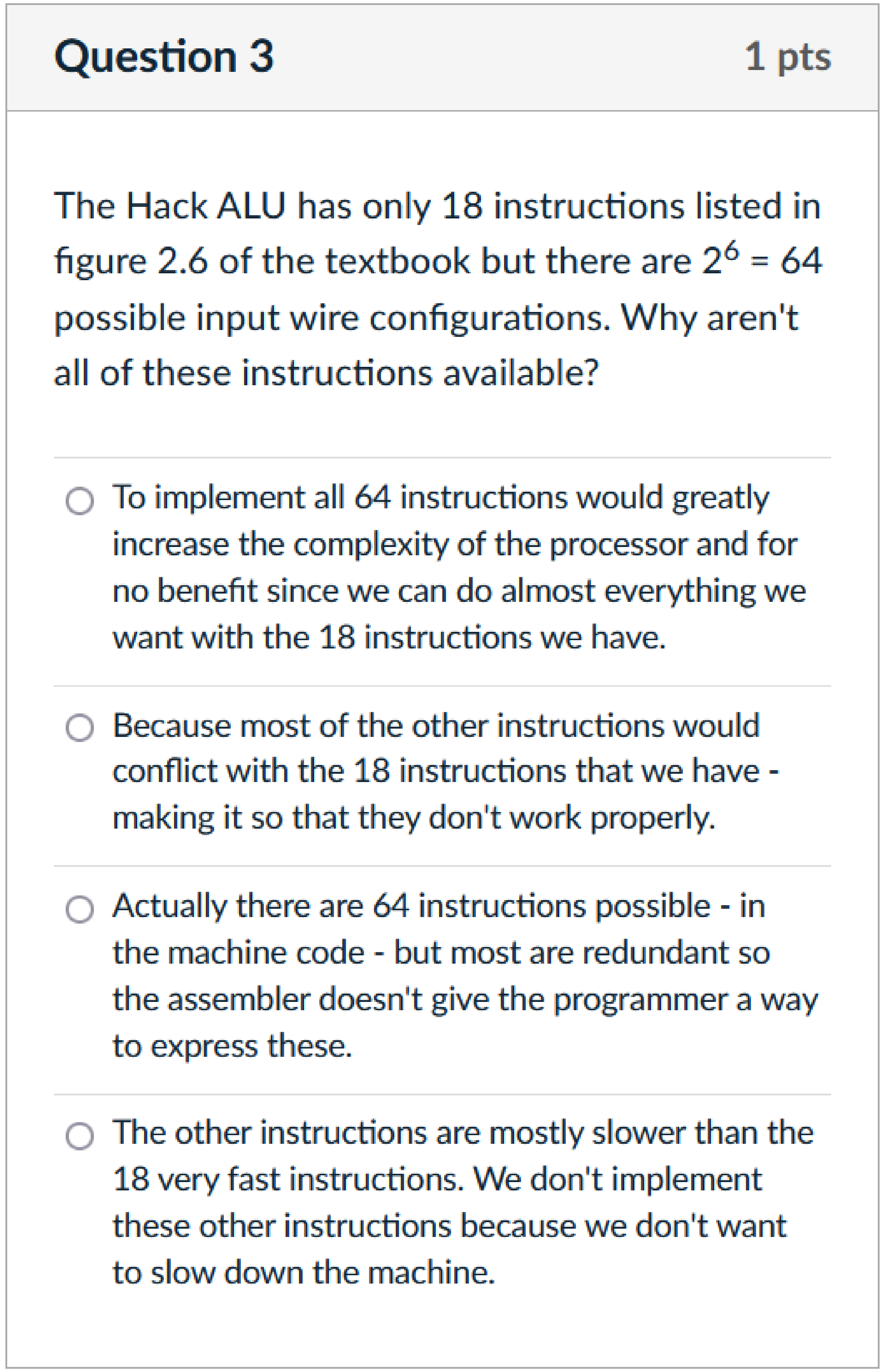
1 pts

Why is the following instruction - which is valid hack assembler - unlikely to be useful when run on the Hack machine?

```
D=M;JMP
```

- ○ The a register gets used and reset by its use in accessing M leaving it set to zero for the jump which returns it to the start of the program which is very rarely what you want.
- ○ Because it accesses the data memory at M using A and then also jumps to that same A address in ROM. It is very unlikely that this address is useful in both memories at the same time.
- ○ Because it implicitly overwrites the A register and then jumps to that overwritten location.
- ○ Because we can't combine jumps with other instructions in C-instructions.

THE UNIVERSITY of ADELAIDE



Question 3

1 pts

The Hack ALU has only 18 instructions listed in figure 2.6 of the textbook but there are $2^6 = 64$ possible input wire configurations. Why aren't all of these instructions available?

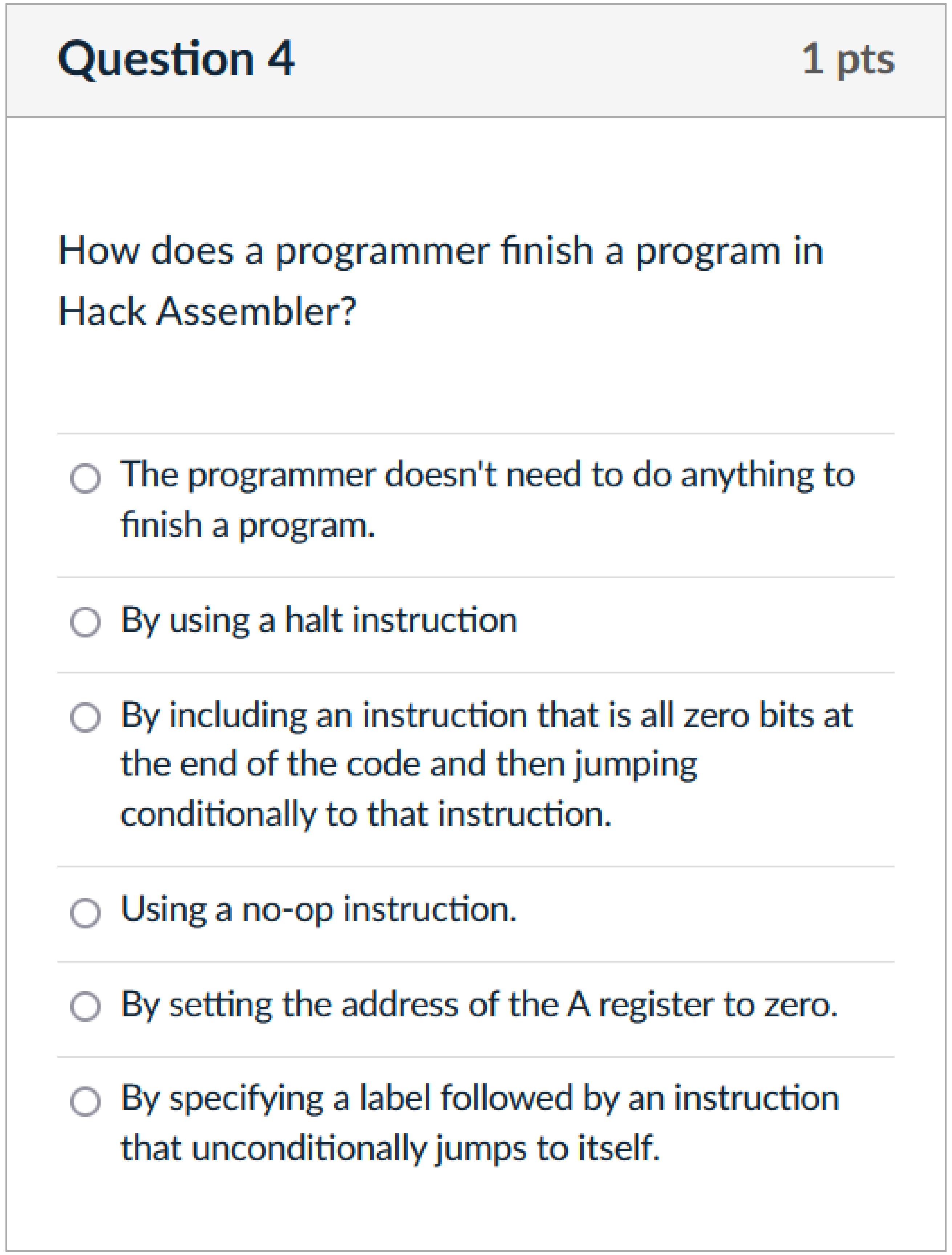
○ To implement all 64 instructions would greatly increase the complexity of the processor and for no benefit since we can do almost everything we want with the 18 instructions we have.

○ Because most of the other instructions would conflict with the 18 instructions that we have - making it so that they don't work properly.

○ Actually there are 64 instructions possible - in the machine code - but most are redundant so the assembler doesn't give the programmer a way to express these.

○ The other instructions are mostly slower than the 18 very fast instructions. We don't implement these other instructions because we don't want to slow down the machine.

THE UNIVERSITY of ADELAIDE

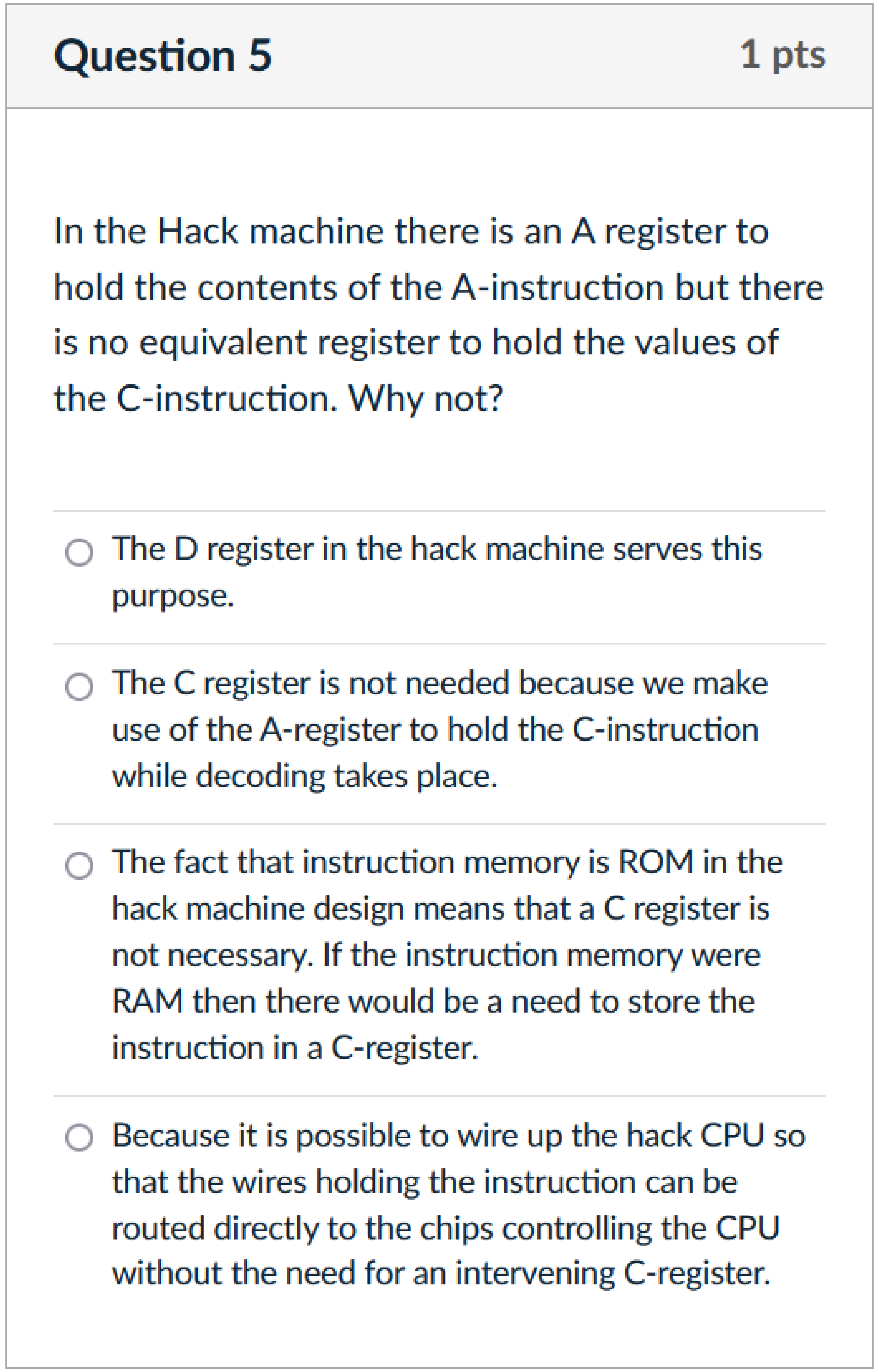


Question 4 **1 pts**

How does a programmer finish a program in Hack Assembler?

- ○ The programmer doesn't need to do anything to finish a program.
- ○ By using a halt instruction
- ○ By including an instruction that is all zero bits at the end of the code and then jumping conditionally to that instruction.
- ○ Using a no-op instruction.
- ○ By setting the address of the A register to zero.
- ○ By specifying a label followed by an instruction that unconditionally jumps to itself.

THE UNIVERSITY *of* ADELAIDE



Question 5

1 pts

In the Hack machine there is an A register to hold the contents of the A-instruction but there is no equivalent register to hold the values of the C-instruction. Why not?

- ○ The D register in the hack machine serves this purpose.
- ○ The C register is not needed because we make use of the A-register to hold the C-instruction while decoding takes place.
- ○ The fact that instruction memory is ROM in the hack machine design means that a C register is not necessary. If the instruction memory were RAM then there would be a need to store the instruction in a C-register.
- ○ Because it is possible to wire up the hack CPU so that the wires holding the instruction can be routed directly to the chips controlling the CPU without the need for an intervening C-register.

THE UNIVERSITY *of* ADELAIDE

The C-instruction

```
dest = x + y
dest = x - y
dest = x
dest = 0
dest = 1
dest = -1
```

x = {A, D, M}

y = {A, D, M , 1}

A/M/D/1 are interchangeable in *x* or *y*

dest = {A, D, M, MD, A, AM, AD, AMD, null}

Must follow the order of **A/M/D** in *dest*

Exercise: In small groups implement the following tasks using Hack :

- Set **D** to **A-1**
- Set both **A** and **D** to **A + 1**
- Set **D** to **19**
- Set both **A** and **D** to **A + D**
- Set **RAM[5034]** to **D - 1**
- Set **RAM[53]** to **171**
- Add **1** to **RAM[7]**, and store the result in **D.**

- ❑Set **D** to **A-1**
- ❑Set both **A** and **D** to **A + 1**
- ❑Set **D** to **19**

THE UNIVERSITY *of* ADELAIDE

- Set **D** to **A-1**

D=A-1

- Set both **A** and **D** to **A + 1**

AD=A+1

- Set **D** to **19**

@19
D=A

- Set both **A** and **D** to **A + D**
- Set **RAM[5034]** to **D - 1**
- Set **RAM[53]** to **171**

THE UNIVERSITY *of* ADELAIDE

- Set both **A** and **D** to **A + D**

```
AD=A+D
```

- Set **RAM[5034]** to **D - 1**

```
@5034
M=D-1
```

- Set **RAM[53]** to **171**

```
@171
D=A
@53
M=D
```

THE UNIVERSITY of ADELAIDE

- Add **1** to **RAM[7],** and store the result in **D.**

❑Add **1** to **RAM[7],** and store the result in **D.**

```
@7
D=1+M
```

The C-instruction

```
dest = x + y
dest = x - y
dest = x
dest = 0
dest = 1
dest = -1
```

x = {A, D, M}

y = {A, D, M , 1}

dest = {A, D, M, MD, A, AM, AD, AMD, null}

Exercise: In small groups, implement the following tasks using Hack:

- **sum = 0**
- **j = j + 1**
- **q = sum + 12 – j**
- **arr[3] = -1**
- **arr[j] = 0**
- **arr[j] = 17**
- **etc.**

Symbol table:

j	3012
sum	4500
q	3812
arr	20561

(All symbols and values are arbitrary examples)

- sum = 0
- j = j + 1
- q = sum + 12 – j

Symbol table:

j	3012
sum	4500
q	3812
arr	20561

- sum = 0

```
@sum //or @4500
M=0
```

- j = j + 1

```
@j //or @3012
M=M+1
```

- q = sum + 12 – j

```
@sum //or @4500
D=M
@12
D=D+A
@j // or @3012
D=D-M
@q // or @3812
M=D
```

Symbol table:

j	3012
sum	4500
q	3812
arr	20561

- arr[3] = -1
- arr[j] = 0
- arr[j] = 17

Symbol table:

j	3012
sum	4500
q	3812
arr	20561

❑arr[3] = -1

```
@arr //or @20561
D=A
@3
A=D+A
M=-1
```

❑arr[j] = 0

```
@j // or @3012
D=M
@arr // or @20561
A=A+D
M=0
```

❑arr[j] = 17

```
@j // or @3012
D=M
@arr // or @20561
D=A+D
@temp // allocated at RAM[16]
M=D
@17
D=A
@ temp
A=M
M=D
```

Symbol table:

j	3012
sum	4500
q	3812
arr	20561

Coding examples

Implement the following tasks using Hack commands:

- goto 50
- if D==0 goto 112
- if D<9 goto 507
- if RAM[12] > 0 goto 50
- if sum>0 goto END
- if x[i]<=0 goto NEXT.

Hack commands:

A-command: @value // set A to value

C-command: dest = comp ; jump // dest = and ;jump
// are optional

Where:

comp = 0 , 1 , -1 , D , A , !D , !A , -D , -A , D+1 ,
A+1 , D-1, A-1 , D+A , D-A , A-D , D&A ,
D|A , M , !M , -M ,M+1, M-1 , D+M , D-M ,
M-D , D&M , D|M

dest = M , D , MD , A , AM , AD , AMD, or null

jump = JGT , JEQ , JGE , JLT , JNE , JLE , JMP, or null

In the command dest = comp; jump, the jump materialdes if (comp jump 0) is true. For example, in D=D+1,JLT, we jump if D+1 < 0.

Hack convention:

- True is represented by -1
- False is represented by 0

Symbol table:

sum	2200
x	4000
i	6151
END	50
NEXT	120

(All symbols and values in are arbitrary examples)

- goto 50
- if D==0 goto 112
- if D<9 goto 507

Symbol table:

sum	2200
x	4000
i	6151
END	50
NEXT	120

- goto 50

```
@50
0;JMP
```

- if D==0 goto 112

```
@112
D;JEQ
```

- if D<9 goto 507

```
@9
D=A-D
@507
D;JGT
```

Symbol table:

sum	2200
x	4000
i	6151
END	50
NEXT	120

- if RAM[12] > 0 goto 50
- if sum>0 goto END
- if x[i]<=0 goto NEXT.

Symbol table:

sum	2200
x	4000
i	6151
END	50
NEXT	120

- if RAM[12] > 0 goto 50

```
@12
D=M
@50
D;JGT
```

- if sum>0 goto END

```
@sum // or @2200
D=M
@END // or @50
D;JGT
```

- if x[i]<=0 goto NEXT.

```
@x // or @4000
D=A
@i // or @6151
A=M
A=D+A
D=M
@NEXT // or @120
D;JLE
```

Symbol table:

sum	2200
x	4000
i	6151
END	50
NEXT	120

This Week

- Review Chapters 4 & 5 of the Text Book (if you haven't already).
- Assignment 3 due next Friday (26 Aug 2022).
- Practical Exam; details in announcement.
  - The mobile coverage is poor in IW B23 where our practical exam take place, you may not be able to receive SMS to login Myuni.
  - Make sure you have Okta or Google Authenticator MFA Setup.
- Read Chapter 6 of the Text Book before next week.

THE UNIVERSITY of ADELAIDE