Question 1

1 / 1 pts

Look at the following (incomplete) diagram of the Hack CPU. Look at the wire pointed to by the large red arrow.

Where does the signal on this wire come from and what action does this signal trigger?

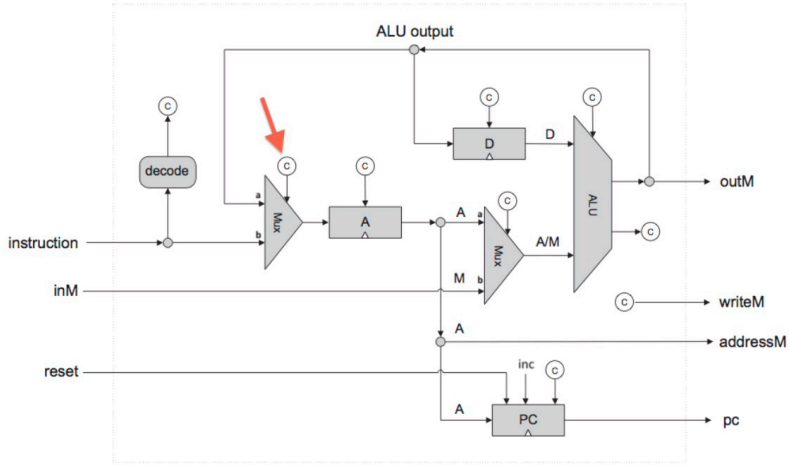

Figure 5.9 Proposed CPU implementation. The diagram shows only *data* and *address paths*, namely, wires that carry data and addresses from one place to another. The diagram does not show the CPU's *control logic*, except for inputs and outputs of control bits, labeled with a circled "c". Thus it should be viewed as an incomplete chip diagram.

The signal on this wire comes from the left-most bit of the instruction (i15)

The signal on this wire comes from the left-most bit of the instruction (i15)

The signal will trigger whether the value routed to the A-register will come from the ALU or an the value in an A-instruction.

Answer 1:

Correct!

the left-most bit of the instruction (i15)

Answer 2:

Correct!

will trigger whether the value routed to the A-register will come from the ALU or an the value in an A-instruction.

Question 2

0 / 1 pts

Look at the following (incomplete) diagram of the Hack CPU. Look at the wire (and it is a single wire) pointed to by the large red arrow.

Where does the signal on this wire come from and what action does this signal trigger?

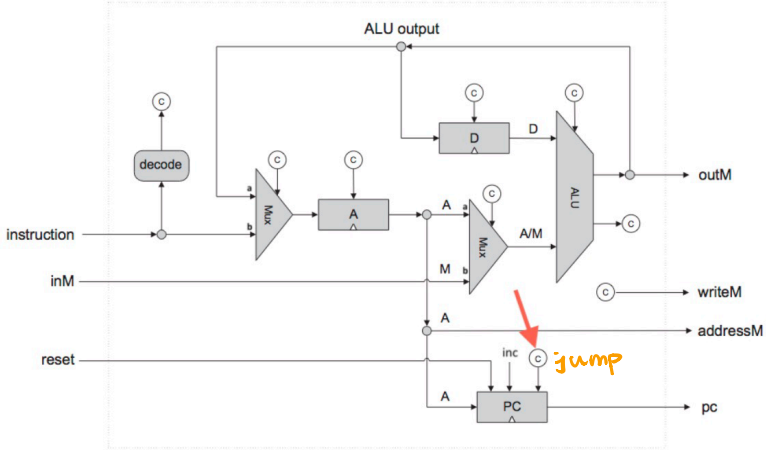

Figure 5.9 Proposed CPU implementation. The diagram shows only *data* and *address paths*, namely, wires that carry data and addresses from one place to another. The diagram does not show the CPU's *control logic*, except for inputs and outputs of control bits, labeled with a circled "c". Thus it should be viewed as an incomplete chip diagram.

The signal on this wire comes from [Select]

The signal [Select]

Answer 1:

Correct answer — the result of a jump calculation (combines the status signals from the ALU with the three jump wires of the C-instruction)

You Answered — the "ng" wire that comes out of the ALU.

Answer 2:

Correct answer — determines whether the program counter should increment to load the next instruction, or jump to the instruction at address A.

You Answered — determines whether the program counter should increment to load the next instruction, or stay the same.

Question 3

1 / 1 pts

What does the following Hack assembler code always do to the current value in register D?

```
D=!D
D=D+1
```

- Sets D to be 0
- Sets D to be 1
- Sets D to be 1 - D
- Correct! Sets D to be -D

This flips all the bits in D and then adds 1. This is how you negate a number in 2s complement.

Question 4

1 / 1 pts

Category: Assembler

Briefly describe what the following Hack assembler code does

```
        @KBD
        D=M <-D=RAM[KBD]
        @48
        D=D-A<-D=RAM[KBD]-48
        @num
        M=D <-RAM[num] = RAM[KBD]-48
(END)
        @END
        0;JMP
```

- Correct! It reads the keyboard code at the time of the first D=M command and subtracts 48 from it and puts the result in "num"
- It reads the keboard register at the time of the @kbd command and exits if the scan code is more than 48.
- Nothing, there is no way that this code is able to read the keyboard unless the user is able to press the number '48' which doesn't exist on the keyboard. If it could work it would put the value -48 in "num".
- Nothing, there is a syntax error and the code doesn't assemble.

Question 5

1 / 1 pts

Category: Assembler

What does the following Hack assembler code do?

```
        @pix
        M=1     RAM[pix]=1
        D=M     D=1
        M=M+D   RAM[pix]=1+1
        M=M+1   RAM[pix]=2+1
        D=M     D=3
        M=M+D   RAM[pix]=3+3=6
        M=M+1   RAM[pix]=7
        D=M     D=7
        @SCREEN
        M=D
(END)
        @END
        0;JMP
```

- It draws the following pixels black, pixel 0, pixel 16 and pixel 32.

Correct!

- It draws the topmost leftmost three pixels on the screen black.
- It draws one pixel black.
- It draws no pixels black because of overflow.