Exercise 1 – The GOTO Instruction

A – in the PC register

IJVM instructions are the ones stored in the Method Area of memory. We find those instructions using the PC register as a pointer. "Main1" is the address of the microinstructions which execute the IJVM instruction we find.

Exercise 2 – MIC-1 Summary

G – B and D above

Answer A is true for most instructions, but not for GOTO and conditional jump instructions.

Answer C is false – the TOS register contains the a copy of the value at the top of the stack, not the address of the top of the stack (the address is in the SP register).

Answer E is false – it's the JAMN and JAMZ bits that are used to perform conditional jumps to different microinstructions JMPC is used to decode the IJVM opcodes by selecting a next microinstruction from the control store address equivalent to the opcodes.

Exercise 3 – Three-Bus Architecture

A – each microinstruction word will need to have more bits to supply the extra control signals that drive the connections to the third bus

With the two-bus architecture, we required a "B Bus" field in the microinstruction to control which register would be connected to the B Bus. In the three-bus architecture we'll need to add an "A Bus" field (four more bits) to control which register is connected to the A Bus.

Answer B is false – in fact with a third bus we'll be able to eliminate some microinstructions because we won't always be needing to move things into the H register.

Answer C is false – a single microinstruction word per clock cycle is still all we need, although the word will have more bits in it.

Exercise 4 – Instruction Fetch Unit

C – The microinstruction references "MBR1", so this pulls one byte from the IFU. As a result, the remaining two bytes are shifted to the left end of the register – this leaves four empty bytes and therefore the IFU automatically fetches the next four bytes from the Method Area.

Exercise 5 – Instruction Fetch Unit

A – The microinstruction does not references "MBR1" or "MBR2", so the state of the instruction fetch unit does not change.

Exercise 6 – Instruction Fetch Unit

D – The microinstruction references "MBR1U" and "MBR1", so two bytes are removed from the IFU. The remaining byte is shifted to the left, which leaves five empty bytes and therefore the IFU automatically fetches the next four bytes from the Method Area.

Exercise 7 – MIC-2 Microcode

C – Last week, the two-bus version of this microcode was:

```
H = TOS
MAR = SP
MDR = TOS = -H; wr;  goto MAIN
```

We had to move the TOS value into H so that we could negate it, since there was no way to connect TOS to the ALU's A input. In the three-bus architecture we can send TOS directly into the A input so we're able to eliminate one microinstruction (two if you count the fact that we no longer need to execute the MAIN microinstruction).

Exercise 8 – MIC-3 Microcode

B – **the microcode is incorrect**

Again, the problem is the trap of trying to negate the "B" input of the ALU – it's not possible.

The microcode should read as follows:

	Load Inputs	ALU Operation	Store Outputs
1	A = TOS		
2	A = SP	C = –A	
3		C = A	MDR = C
4			MAR = C; wr; goto (MBR1)