HW #6

1. a) 1001 0000 1110 0110 (E, 6) ADD M[M[E6]]

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

As the 15 bit is 1 and the op-code 001, the instruction is an indirect address for a memory reference instruction. So the instruction will go to a specific address and store the actual operand to be executed. In particular, this instruction will ADD the contents of the address (EC) to the AC.

b) 0011 1001 1100 1010 (a, C, A) STA M[9CA]

This instruction is a memory reference operation using direct addressing. It will store the contents of address specified by bits 0-11 (0_C-BB_C) into the AC.

2.

	E	AC	PC	AR	IR
Initial	0	30AA	073	—	—
CLA	1	0000	074	300	7800

3. PC = 119
AC = 30
M[300] = 600
M[600] = the

First we branch to the actual instruction so the PC will be loaded to the AR.

PC = AR = 119

AR ← PC, then the PC is incremented and we fetch the memory address to put into IR

an indirect address

IR = 9300 PC = 120

IR ← M[AR], PC ← PC+1

fetch

This is now ready to see what the instruction is. We get the bits of IR and see that the first bit is 1 in the decoder and all other outputs are set to 0. This means this instruction is a memory-reference instruction.

$D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7$,
0 1 0 0 0 0 0 0

So then we get the actual address of the operand (M[600] in this case) and load it to the AR. IR = 300 = AR

AR ← IR (first 12 bits starting from the right)

AR ← M[AR] AR = 300

Now we can execute the instruction. Load the instruction to the Data Register. Add the DR and AC together and trigger E flip flop to 1.

DR ← M[AR] DR = FF

AC ← DR + AC AC = 1

E ← Cout E = 1

b) Since the computer is in the process of the fetch cycle, the fetch cycle runs and then the computer branches to the interrupt cycle, where the current PC is cleared to zero address 0.

AR ← 0, TR ← PC M[0] = 119

M[AR] ← TR, PC ← 0

The computer then fetches the first instruction of the interrupt subroutine as part of the next instruction cycle.