1. **A)** Since we know that we have a memory unit with 20 bits per word and all instructions have an opcode and a single address (like MARIE), we know that the address is supposed to be 12. So, the opcode is what remains which is 20-12. Final result: **8.**

   B) Since there are 20 bits in total and 8 are for the opcode, there are 12 bits left for the address. Final result: **12.**

   C) The maximum allowable size for memory is 2n, where n is the number of address lines. Therefore, we have 2^{12} = 4096. Final result: **4096 bytes is the maximum allowable size for memory.**

   D) The highest value is (2^n-1). Therefore, $(2^{12}-1)$= 4095. And in binary,4095 is equal to 111111111111. Final result:**111111111111 is the largest unsigned binary integer that can be stored in one word of this memory.**

2. **Human readable mnemonic form of 5000, 6000, 4127.**

   5000: Opcode is 5 (input), thus input a value from the keyboard to a AC.

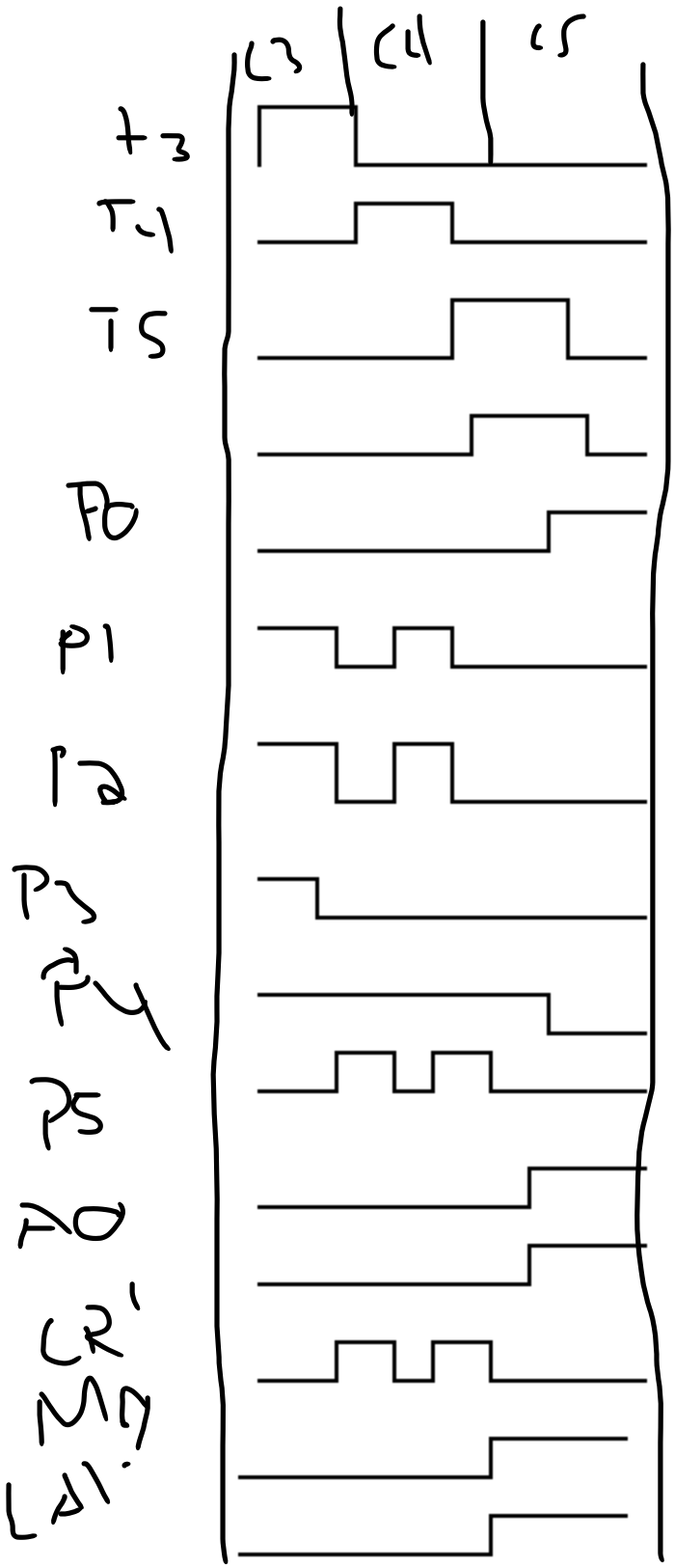
   6000: Opcode is 6 (output), thus output the value in the AC to the display.

   4127: Opcode is 4 (subtract), thus subtract the contents of the address 127 from the AC.

   In conclusion, our assembly code is

   ```
   Input
   Output
   Subt[127]
   ```

3. **Timing Diagram**



4. A) Since we're checking for Big Endian we read from up to down

3F041B44

Then we convert the hex to binary and we get:

00111111000001000001101101000100
We do not need to take the 2's complement because the number is positive.

And from binary to decimal and we get:

1057233732

Final result: $\mathbf{(1057233732)_{10}}$

B) Since we're checking for the little endian format, we now read from down upwards.

The hex value: 441B043F

Convert the hex to binary: 01000100000110110000010000111111

Since the number is positive we don't need to take 2's complement

Exponent: 136-127= 9 which in binary is 1001

Mantissa: 00110110000010000111111 which in decimal is 0.2110670804977419921

Final result: $\mathbf{0.620066345215 \times 10^{3}}$

5. A) Convert 427*32-*3-+ to infix

4	4
2	4,2
7	4,2,7
*****	4 (2*7)
3	4 (2*7),3
2	4 (2*7),3,2
-	4 (2*7),3,2,-
*****	4 (2*7),(3-2),*
3	4 (2*7)*(3-2),3
-	4 (2*7)*(3-2),3,-
+	4 (2*7)*(3-2)-3,+

Final Result: **4+(2*7)*(3-2)-3**

B) Convert (9-3+2*(4+3-4))/(4-2*3) to postfix

Expression	**Stack**	**Postfix**
	(	
9	((	9
	((-	9
3	((-	93
+	((+	93-
2	((+	93-2
*****	((+*	93-2
4	((+*(	93-24
+	((+*(+	93-24
3	((+*(+	93-243
-	((+*(-	93-243+
4	((+*(-	93-243+4
)	((+*	93-243+4-
)	(	93-243+4-*+
/	(/	93-243+4-*+
4	(/(	93-243+4-*+4
	(/(-	93-243+4-*+4
2	(/(-	93-243+4-*+42

*	(/(-*	93-243+4-*+42
3	(/(-*	93-243+4-*+423
)	(/	93-243+4-*+423*-
)		93-243+4-*+423*-/

Final result: 93-243+4-*+423*-/

6. **A) The answer is the value 400 because in the immediate addressing mode the operand contains actual value to be used.**

   B) The value loaded will be 800 because the operand states the location of the value to be used.

   C) The value loaded will be 300 because the operand is the address of the address of the data. So, load 400 will load 800 and the data of 800 is 300.

   D) The value loaded will be 200 because the register is 300 so it'll be (R1+400)which is load 700 and the accumulator will load its value which is 200.

7. Assembly source file included in zip file