Word Bank - questions 1-8. Place the appropriate term by its description below. (2 pts. each)

ALU     load    direct    instruction
location counter (loc)    MAR    store    immediate
pseudo-op    program counter (PC)    MDR    branch
opcode    assembler    instruction register (IR)

1. __________ a symbolic name in an assembly language program used to access data or as a branch target

2. __________ a number used as the name of a memory location

3. __________ a variable in the assembler program that contains the address at which the next instruction or data word will be placed

4. __________ the basic unit of work in a program, which is composed of an opcode and possibly one or more operands

5. __________ an assembly language statement that doesn't generate a machine instruction but instead acts as a directive to the assembler

6. __________ a functional unit within a computer that performs arithmetic and logical computations

7. __________ a register in the bus interface unit that contains the value that has been read from or should be written to memory

8. __________ an addressing mode in which the instruction holds the address of the operand

Give the power of ten that corresponds to these prefixes. (2 pts. each)

9. giga ________

10. peta ________

11. mega ________

12. tera ________

13. Define "assembly language" and "machine language", clearly indicating both the difference between them and the relationship between them. (12 pts.)
14. On the right hand side, give the symbol table produced by the first pass of the assembler for the following accumulator machine program. (16 pts.)

15. The opcodes for the load, add, store and halt instructions are 50, 40, 60, and 0, respectively. On the right hand side give the executable file produced by the second pass of the assembler for the following accumulator machine program. (16 pts.)

```
label(constants)
  word(two, 2)
  word(one, 1)
  word(zero, 0)

label(variables)
  word(x, 0)

label(main)
  load(zero)
  add(one)
  add(two)
  store(x)
  halt

end(main)
```
accumulator machine instruction set

<table>
<thead>
<tr>
<th>opcode</th>
<th>address</th>
<th>operation name</th>
<th>machine action</th>
</tr>
</thead>
<tbody>
<tr>
<td>halt</td>
<td>----</td>
<td>halt</td>
<td>stop execution</td>
</tr>
<tr>
<td>div</td>
<td>addr</td>
<td>divide</td>
<td>acc = acc/memory[addr]</td>
</tr>
<tr>
<td>mul</td>
<td>addr</td>
<td>multiply</td>
<td>acc = acc*memory[addr]</td>
</tr>
<tr>
<td>sub</td>
<td>addr</td>
<td>subtract</td>
<td>acc = acc-memory[addr]</td>
</tr>
<tr>
<td>add</td>
<td>addr</td>
<td>add</td>
<td>acc = acc+memory[addr]</td>
</tr>
<tr>
<td>load</td>
<td>addr</td>
<td>load</td>
<td>acc = memory[addr]</td>
</tr>
<tr>
<td>store</td>
<td>addr</td>
<td>store</td>
<td>memory[addr] = acc</td>
</tr>
<tr>
<td>ba</td>
<td>addr</td>
<td>branch always</td>
<td>pc = addr</td>
</tr>
<tr>
<td>blt0</td>
<td>addr</td>
<td>branch on less than</td>
<td>if acc&lt;0 then pc = addr</td>
</tr>
<tr>
<td>ble0</td>
<td>addr</td>
<td>branch on less than or equal</td>
<td>if acc&lt;=0 then pc = addr</td>
</tr>
<tr>
<td>beq0</td>
<td>addr</td>
<td>branch on equal</td>
<td>if acc==0 then pc = addr</td>
</tr>
<tr>
<td>bne0</td>
<td>addr</td>
<td>branch on not equal</td>
<td>if acc/=0 then pc = addr</td>
</tr>
<tr>
<td>bge0</td>
<td>addr</td>
<td>branch on greater than or equal</td>
<td>if acc&gt;=0 then pc = addr</td>
</tr>
<tr>
<td>bgt0</td>
<td>addr</td>
<td>branch on greater than</td>
<td>if acc&gt;0 then pc = addr</td>
</tr>
<tr>
<td>print</td>
<td>addr</td>
<td>print</td>
<td>print contents of memory[addr]</td>
</tr>
</tbody>
</table>

16. Write code that implements the following pseudo-code in assembly language for the accumulator machine. Do not simplify. (16 pts.)

    a = 5;
    Assume this data section:
    b = 0;
    word(a,0)
    while( a > 1 ){          word(b,1)
        word(b,1)
        word(zero,0)
        b = b + a;
        word(one,1)
        a = a - 1;
    }                        word(five,5)
    comment(`add others if needed')