Word Bank - questions 1-10. Place the letter of the matching term in the space provided.

A. accumulator machine  I. instruction  Q. memory
B. address  J. instruction register  R. opcode
C. assembler  K. label  S. pseudo-op
D. ALU  L. load  T. program counter (PC)
E. CPU  M. load/store machine  U. stack machine
F. direct addressing  N. location counter (loc)  V. store
G. immediate addressing  O. MAR
H. indirect addressing  P. MDR

K  1. a symbolic name in an assembly language program that is used to access data or as a branch target
S  2. an assembly language statement that doesn’t generate a machine instruction but instead acts as a directive to the assembler
L  3. an instruction that will make a copy of the value in a specified memory word and place it into the accumulator (or other register)
T  4. a register in the CPU that contains the address of the memory word from which the next instruction will be fetched
D  5. the unit within a computer that performs the arithmetic and logical computations (it has no registers)
M  6. a machine design that contains a register file and for which the arithmetic and logic instructions identify these registers as sources and destinations rather than memory locations
R  7. a mnemonic name in an assembly language program or a number in an executable that identifies the operation
I  8. the basic unit of work in a program, which is composed of an opcode and possibly one or more operands.
O  9. a register in the bus interface unit that contains the address of the memory word that should be read or written.
F  10. addressing mode in which the instruction holds the address of a memory word which contains the data which is to be accessed

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

11. nano \(10^{-9}\)  13. milli \(10^{-3}\)
12. tera \(10^{12}\)  14. micro \(10^{6}\)

Extra Credit 1: (a) With whom did the idea of the stored program concept originate? (2 pts)
(b) To whom is it credited?

a. Mauchly and Eckert
b. Von Neumann
15. For the accumulator machine, the load instruction operated as follows:

\[
\text{load(x)} \Rightarrow \text{"acc = memory[x];"}
\]

Suppose we want to implement a load indirect instruction as follows:

\[
\text{loadi(x)} \Rightarrow \text{"acc = memory[memory[x]];"}
\]

For the following data and code, fill in the data value for place and below that fill in the trace of the accumulator contents. The data starts at address 0. (10 pts.)

\[
\begin{align*}
\text{word(five,5)} & \quad \text{// symbolic location "five" contains} \quad 5 \\
\text{word(ten,10)} & \quad \text{// symbolic location "ten" contains} \quad 10 \\
\text{word(place,ten)} & \quad \text{// symbolic location "place" contains} \quad 1
\end{align*}
\]

label(start)
\[
\begin{align*}
\text{load(ten)} & \quad \text{// acc = } 10 \\
\text{sub(ten)} & \quad \text{// acc = } 0 \\
\text{load(place)} & \quad \text{// acc = } 1 \\
\text{loadi(place)} & \quad \text{// acc = } 10 \\
\text{halt}
\end{align*}
\]

16. On the right hand side, give the symbol table produced by the first pass of the assembler for the following accumulator machine program. (10 pts.)

\[
\begin{align*}
\text{word(a,10)} \\
\text{word(b,25)} \\
\text{word(c, 0)} \\
\text{label(alpha)} \\
\text{load(a)} \\
\text{sub(b)} \\
\text{store(c)} \\
\text{bge0(finished)} \\
\text{sub(c)} \\
\text{sub (c)} \\
\text{store(c)} \\
\text{label(finish ed)} \\
\text{halt} \\
\text{end(alpha)}
\end{align*}
\]

\[
\begin{array}{|c|c|}
\hline
\text{Symbol} & \text{Address} \\
\hline
a & 0 \\
b & 1 \\
c & 2 \\
alpha & 3 \\
finished & 17 \\
\hline
\end{array}
\]

17. Is the reference to "a" in the load instruction above a forward or backward reference? (2 pts.)

backward
18. The opcodes for the halt, load, sub, store, and bge0 instructions are 0, 50, 30, 60, and 75, respectively. On the right hand side give the executable file produced by the second pass of the assembler for the following accumulator machine program. (15 pts.)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>word(a,10)</td>
<td>10</td>
</tr>
<tr>
<td>word(b,25)</td>
<td>25</td>
</tr>
<tr>
<td>word(c, 0)</td>
<td>0</td>
</tr>
<tr>
<td>label(alpha)</td>
<td>50 0</td>
</tr>
<tr>
<td>load(a)</td>
<td>30 1</td>
</tr>
<tr>
<td>sub(b)</td>
<td>60 2</td>
</tr>
<tr>
<td>store(c)</td>
<td>75 17</td>
</tr>
<tr>
<td>bge0(finished)</td>
<td>30 2</td>
</tr>
<tr>
<td>sub(c)</td>
<td>30 2</td>
</tr>
<tr>
<td>sub(c)</td>
<td>60 2</td>
</tr>
<tr>
<td>store(c)</td>
<td>0</td>
</tr>
<tr>
<td>label(finished)</td>
<td></td>
</tr>
<tr>
<td>halt end(alpha)</td>
<td>3</td>
</tr>
</tbody>
</table>
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>display contents of memory[addr]</td>
</tr>
</tbody>
</table>

19. Write code that implements the following program segment in assembly language for the accumulator machine. (20 pts.) Write your answer on the paper provided.

```assembly
value = 1;
while(value <= 10 ){
    value = 2*value;
}

word(one, 1)
word(two,2)
word(ten,10)
word(value,0)

label(start)
load(one)
store(value)

label(loop)
sub(ten)
bgt0(done) comment('if i-5 > 0 jump out of loop')
load(value)
mul(two)
store(value)
ba(loop)
label(done)
) halt
end(start)
```

Assume this data section:

- `value` is initialized to 1.
- The loop continues as long as `value` is less than or equal to 10.
- The value is doubled in each iteration.
- The data section includes the following values:
  - `one`
  - `two`
  - `ten`
  - `value`

Note: You can add other data as needed.
Extra credit 2. (up to 8 pts.)
Write a section of code in assembly language for the accumulator machine that implements the following loop.
Write your answer on the paper provided.
for( i = 1; i <= 5; i++ ) print(i);

    word(one,1)
    word(five,5)
    word(i,0)
label(start)
    load(one)
    store(i)
label(loop)
    load(i)
    sub(five)
    bgt0(done) comment(`if i-5 > 0 jump out of loop')
    load(i)
    print(i)
    add(one)
    store(i)

    ba(loop)
    label(done)