A. (25 points) Consider 0-address machine instruction set below:

|  |
| --- |
| **0-address machine (reverse Polish)** |
| where the processor has a stack and some supporting hardware, at least a top of stack (TOS) pointer.  |
| **Operation** | **e.g. or comment**  |
| load\_literal <int>  | effect:TOS:=TOS+1; stack[TOS]:=<int>load a *constant* onto the top of stack; this can be used in arithmetic or to get an address onto the stack for use by a load or a store instruction later (it is splitting hairs to argue whether the literal is an address or a constant which might happen to be used as an address elsewhere)  |
| load  | effect:stack[TOS]:=memory[stack[TOS]]take the top-of-stack as an address, replace the top-of-stack with the contents of that address.  |
| sto  | effect:memory[stack[TOS-1]]:=stack[TOS]; TOS:=TOS-2store contents of top of stack at the address in stack[TOS-1] then pop the value and the address  |
| <opcd>  | where <opcd> is add | sub |...effect:stack[TOS-1] := stack[TOS-1] <op> stack[TOS];TOS:=TOS-1 |

For the expression

y=z/y+w\*(z+y-w)

1. Give the reverse Polish notation (5 points)

Expression tree:

Reverse Polish notation is obtained by Left-Right-Node (LRN) traversal <https://en.wikipedia.org/wiki/Tree_traversal#Post-order,_LRN> of the tree staring from the root:

Yzy/wzy+w-\*+=

1. Write the code to implement it (10 points)

Scan RPN from left to right, and write out commands of placing in the stack address of the left-hand side (LHS) variable values for the right-hand side of the expression (RHS), and operation commands:

1. Load\_literal @y
2. Load\_literal @z
3. Load
4. Load\_literal @y
5. Load
6. Div
7. Load\_literal @w
8. Load
9. Load\_literal @z
10. Load
11. Load\_literal @y
12. Load
13. Add
14. Load\_literal @w
15. Load
16. Sub
17. Mult
18. Add
19. sto

B. (13 points) Use PLA to create a ROM with 9 memory cells containing the following numbers in the given order: 12, 9, 1, 8, 3, 6, 10, 4, 2. For each number, specify its address in your ROM. Each memory cell shall have one and the same number of bits, and this number of bits shall be minimal possible to accommodate the specified nine numbers. What is the number of bits of each memory cell? Why?



Minimal number of bits to represent the maximal number in the set is 4, hence, each cell shall have at least 4 bits (z3, z2, z1, z0). To represent the given nine numbers we need at least 4 address bits (x3, x2, x1, x0).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| address | X3 | X2 | X1 | X0 | Z3 | Z2 | Z1 | Z0 | value |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 12 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 9 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 8 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 3 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 6 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 10 |
| 7 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 4 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |

&0

&1

&2

&6

&7

&8

X3

X3’

X2

X2’

X1

X1’

X0

X0’

&3

&4

&5

Z0

V

Z1

V

Z2

V

Z3

V

