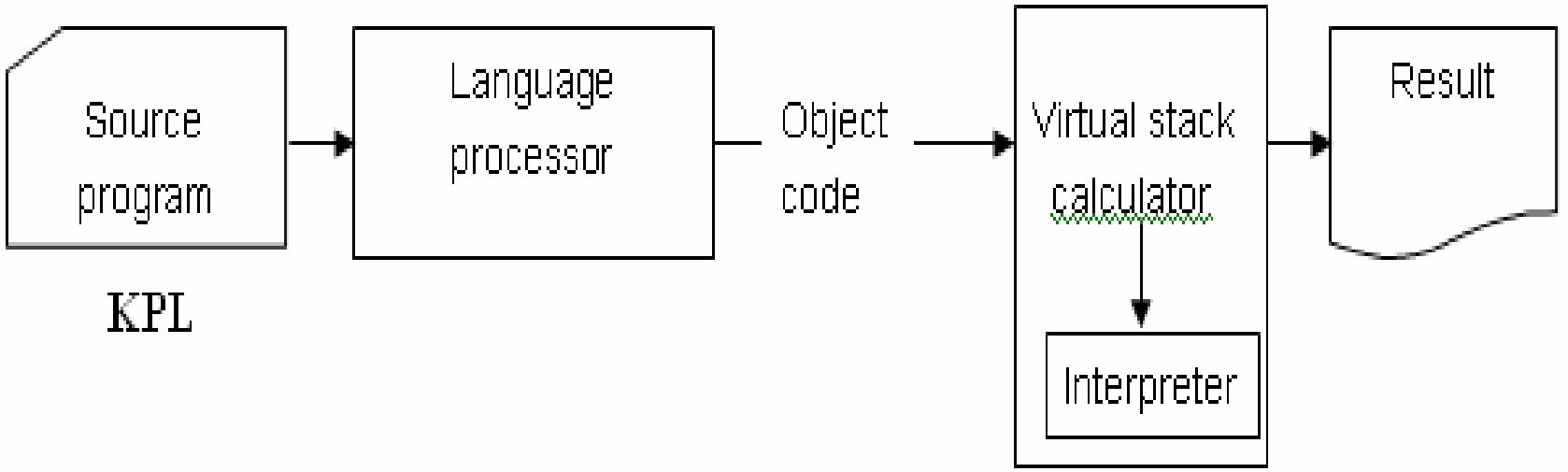
Unit 13

Code Generation

Nguyen Thi Thu Huong
Hanoi University of Technology

Program execution by KPL



Input of Code Generator

- Immediate code
- Source code

Output of Code Generator

- Machine (executable) code
- Assembly code
- ***Intermediate code for a virtual machine***

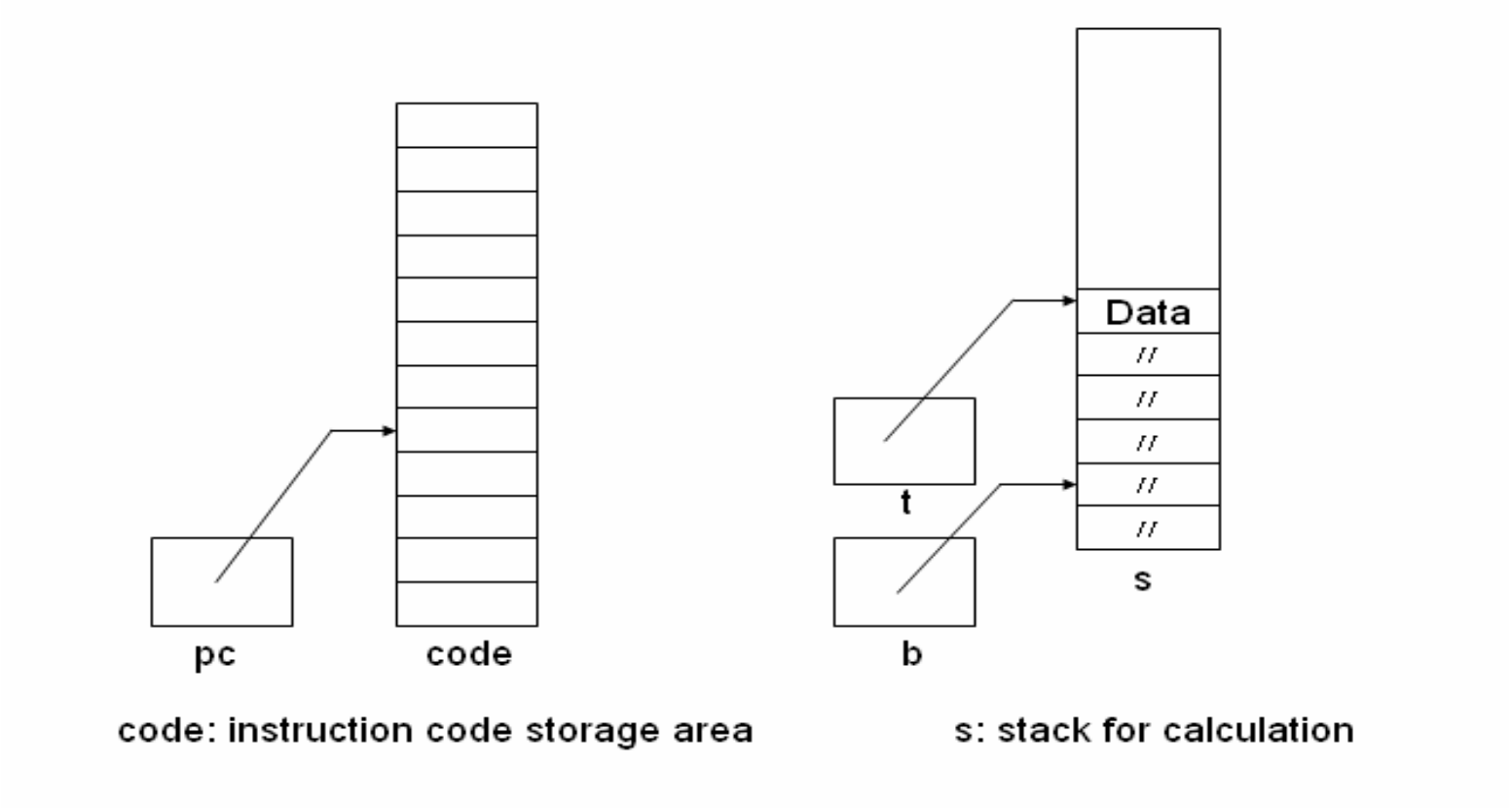
Stack calculator

- Memory area to store instruction codes
- Stack calculator for operation
- Four kinds of register including

Registers

- pc: program counter
- t: stack top
- b: base address of data area on the stack for active block
- ps: status of active program

Structure of Stack Calculator



The status of ‘ps’

- 0: active
- 1: normal end (end with Halt instruction)
- 2: abnormal end due to devide error
- 3: abnormal end due to stack overflow
- 4: abnormal end due to input error
- 5: abnormal end due to output error

Contents of instruction words for Stack calculator

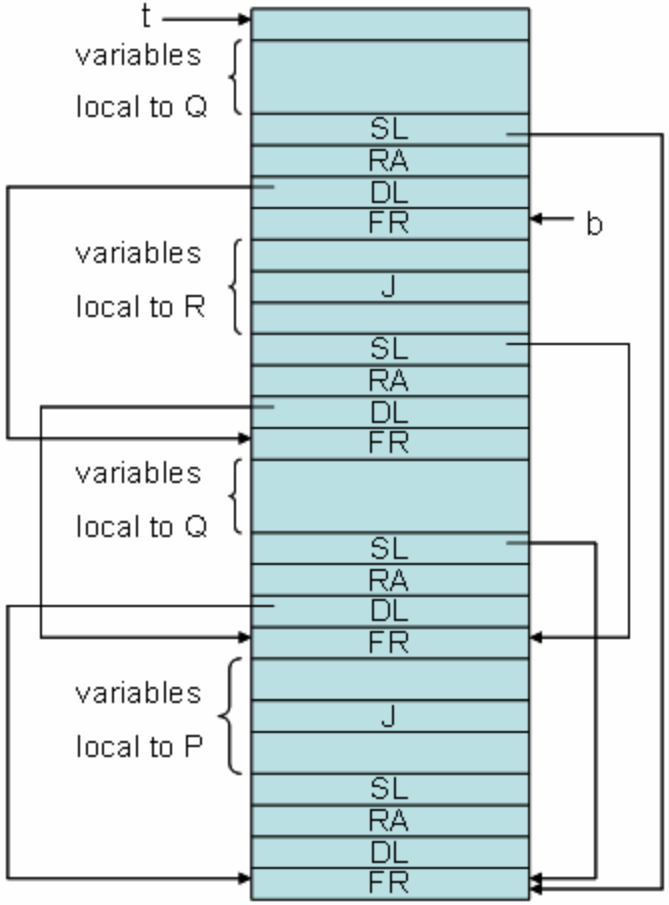
Instruction code	Instruction word	Content of instruction word [= (,] =)
LA	Load Address	t:=t+1; s[t]:=base(p)+q;
LV	Load Value	t:=t+1; s[t]:=s[base(p)+q];
LC	Load Constant	t:=t+1; s[t]:=q;
LI	Load Indirect	s[t]:=s[s[t]];
INT	Increment T	t:=t+q;
DCT	Decrement T	t:=t-q;
J	Jump	pc:=q;
FJ	False Jump	if s[t]=0 then pc:=q; t:=t-1;
HL	Halt	Halt
ST	Store	s[s[t-1]]:=s[t]; t:=t-2;
CALL	Call	s[t+2]:=b; s[t+3]:=pc; s[t+4]:=base(p); b:=t+1; pc:=q;
EP	Exit Procedure	t:=b-1; pc:s[b+2]; b:=s[b+1];
EF	Exit Function	t:=b; pc:=s[b+2]; b:=s[b+1];
RC	Read Character	read one character into s[s[t]]; t:=t-1;
RI	Read Integer	read integer to s[s[t]]; t:=t-1;

Contents of instruction words for Stack calculator (cont)

Instruction code	Instruction word	Content of instruction word [= (,] =)
WRC	Write Character	write one character from s[t]; t:=t-1;
WRI	Write Integer	write integer from s[t]; t:=t-1;
WLN	New Line	CR & LF
AD	Add	t:=t-1; s[t]:=s[t]+s[t+1];
SB	Subtract	t:=t-1; s[t]:=s[t]-s[t+1];
ML	Multiply	t:=t-1; s[t]:=s[t]*s[t+1];
DV	Divide	t:=t-1; s[t]:=s[t]/s[t+1];
NEG	Negative	s[t]:=-s[t];
CV	Copy Top of Stack	s[t+1]:=s[t]; t:=t+1;
EQ	Equal	t:=t-1; if s[t]=s[t+1] then s[t]:=1 else s[t]:=0;
NE	Not Equal	t:=t-1; if s[t]<>s[t+1] then s[t]:=1 else s[t]:=0;
GT	Greater Than	t:=t-1; if s[t]>s[t+1] then s[t]:=1 else s[t]:=0;
LT	Less Than	t:=t-1; if s[t]<s[t+1] then s[t]:=1 else s[t]:=0;
GE	Greater or Equal	t:=t-1; if s[t]>=s[t+1] then s[t]:=1 else s[t]:=0;
LE	Less or Equal	t:=t-1; if s[t]<=s[t+1] then s[t]:=1 else s[t]:=0;

From source code. . .

```
PROGRAM   P;
     VAR   J:INTEGER;

     PROCEDURE   Q;

PROCEDURE   R;
          VAR   J:INTEGER;
          BEGIN   (* R *)

               CALL   Q;

          END   (* R *);

       BEGIN   (* Q *)

            CALL   R;  J:=J+1;

       END   (* Q *);

   BEGIN   (* P *)

      CALL   Q;
```



Object code of typical KPL statements

- Assign statement
- IF statement
- FOR statement
- WHILE statement
- Array treatment

Assign statements A:=E;

LA A

code for (E)

ST

A:identifier

E: expression

IF statement (1): IF C THEN S;

```
        code for (C)
        FJ    L1
        code for (S)
L1:
```

C:condition

S:statement

IF statement (2):IF C THEN S1 ELSE S2;

```
        code for (C)
        FJ    L1
        code for (S1)          C:condition
        J     L2               S1, S2:statement
L1:  code for (S2)
L2:
```

FOR statement: FOR I:=A TO B DO S;

```
     LA    I ---------   When 'I' is a local variable.
     code for (A)        If 'I' is a parameter:
     ST                  LV (address of 'I' )
     code for (B)        Other codes must be changed.
L1:  CV
     LV    I
     GT
     FJ    L2
     code for (S)              I:identifier
     LA    I             A, B:expression
     LV    I             S:statement
     LC    I
     AD
     ST
     J     L1
L2:  DCT   1
```

WHILE statement:WHILE C DO S;

```
L1:  code for (C)
     FJ    L2
     code for (S)
     J     L1
L2:
```

C:condition

S:statement

Treatment of ARRAY

For VAR A:ARRAY(.Ih.) OF ARRAY(.Jh.) OF INTEGER;
the address of A(.I.)(.J.) is A11+(I-1)*elsize1+(J-1)*elsize2
For this, elsize2=1, elsize1=Jh*elsize2

```
LA     A11
code for (I)
LC     1
SB
LC     elsize1              I, J:expression
ML
AD
code for (J)
LC     1
SB
LC     elsize2
ML
AD
```

Call for PROCEDURE

P(VAR N:INTEGER; :INTEGER);

CALL P(I, E)

INT 4

LA I

code for (E)

DCT 4 + Number of parameters I:identirier

CALL P E:expression

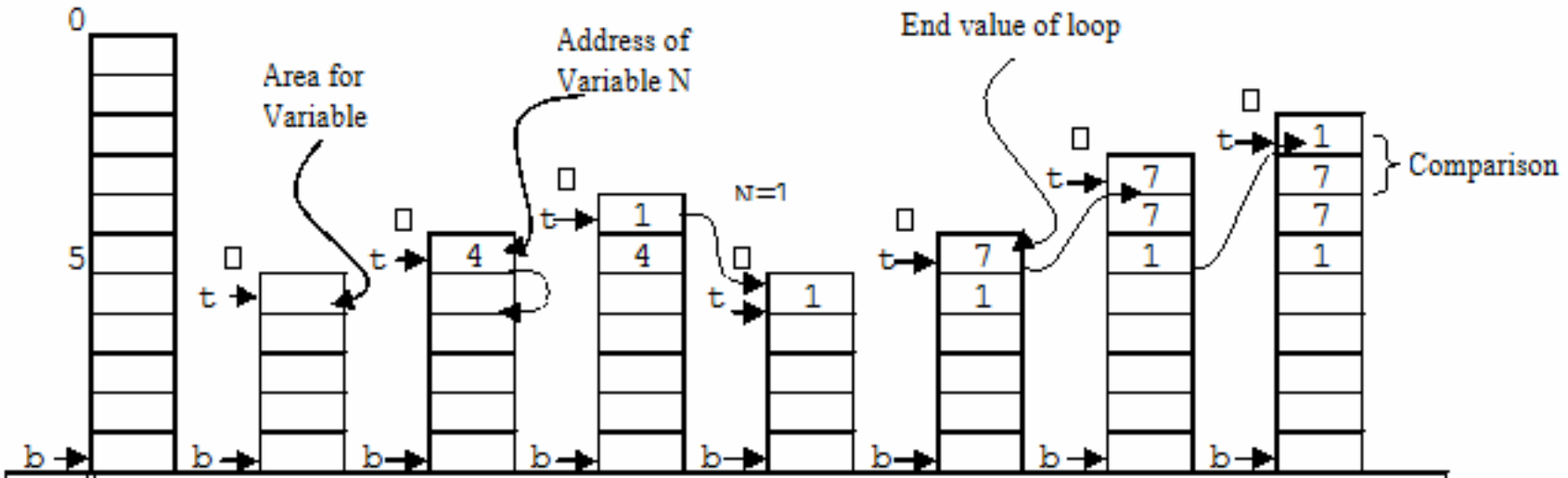
Body of PROCEDURE P

- Source Code

```
PROCEDURE P;
    BEGIN
        S;
    END (* P *);
```

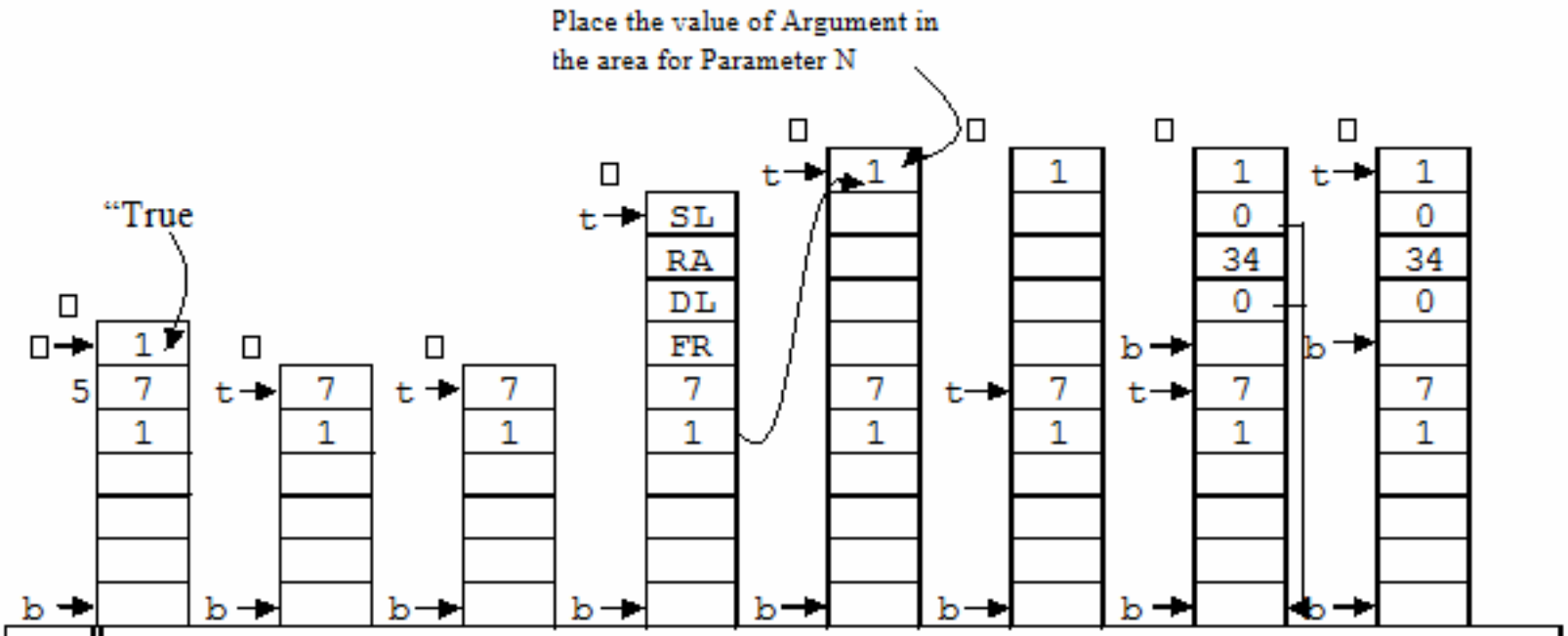
- Object Code

```
INT    4 + Number of parameters + Size of local area
       code for (S); S: statement
       EP
```

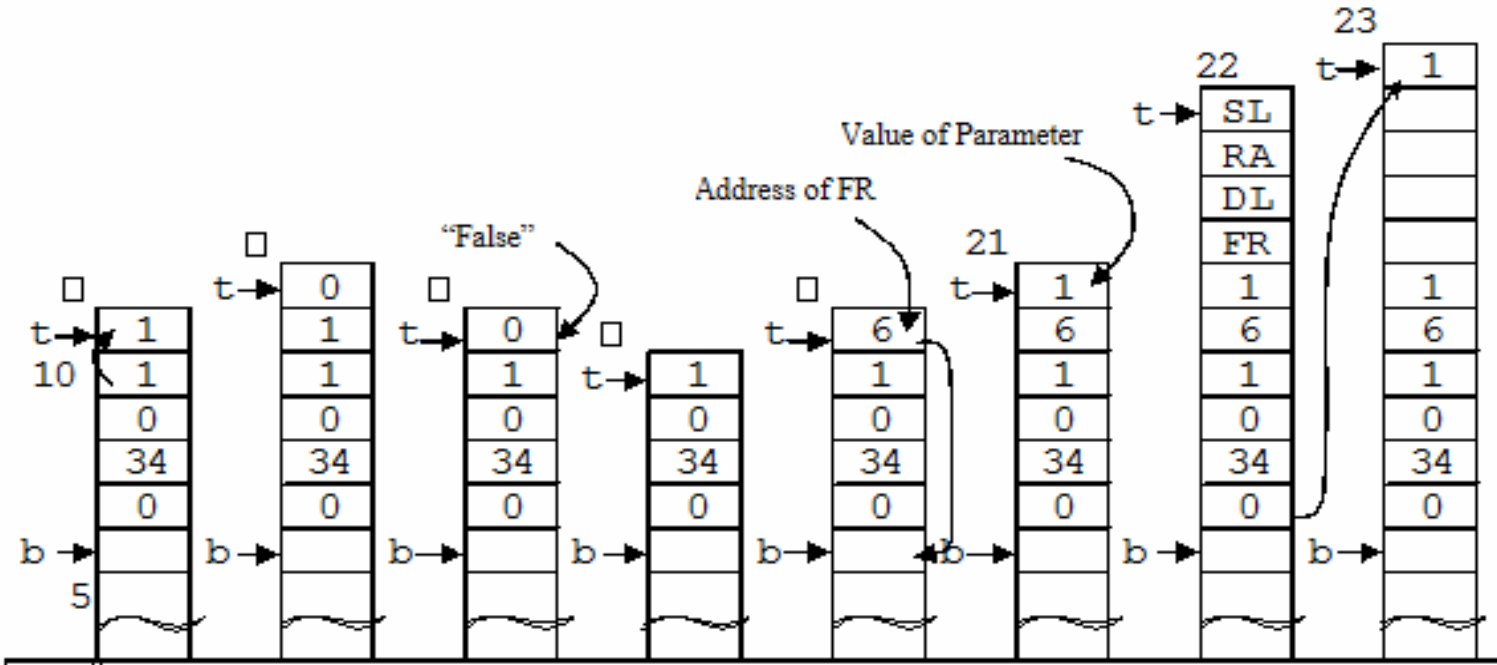


t	-1	4	5	6	4	5	6	7
b	0	0	0	0	0	0	0	0
pc	20	21	22	23	24	25	26	27

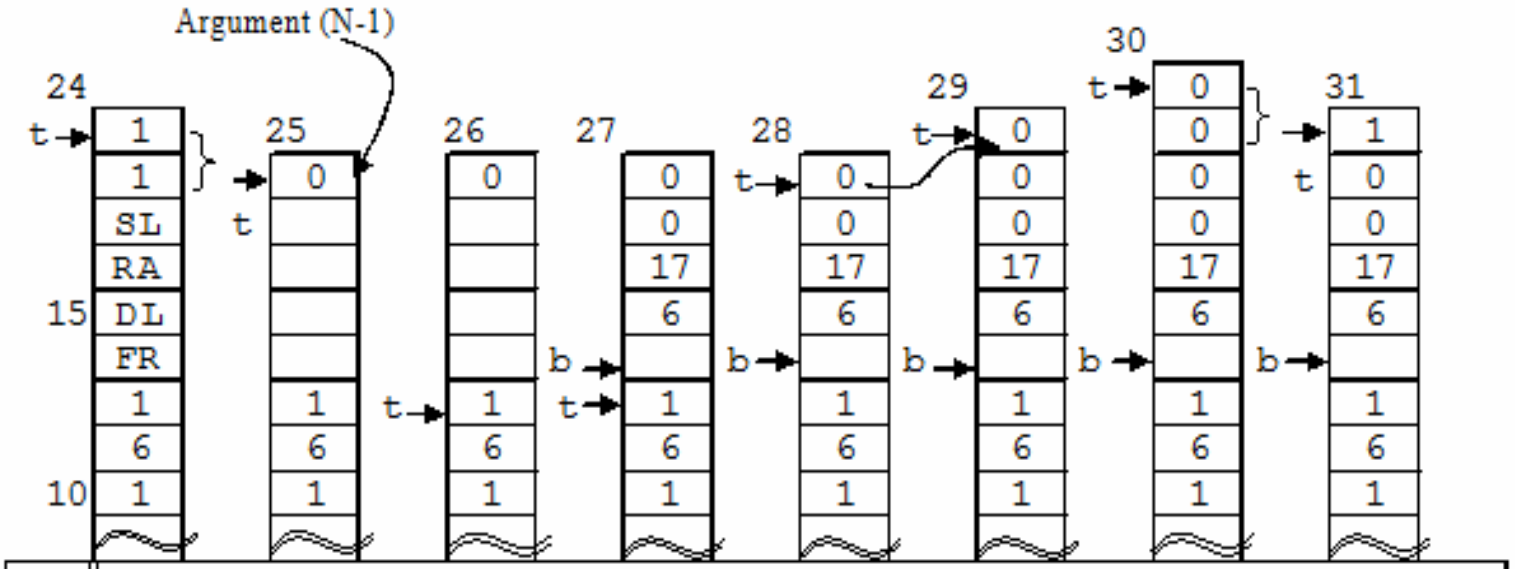
op	INT	LA	LC	ST	LC	CV	LV	GE
pc	0	0	0	0	0	0	0	0
q	5	4	1	0	7	0	4	0



t	6	5	5	9	10	5	5	10
b	0	0	0	0	0	0	6	6
pc	28	29	30	31	32	33	0	1
op	FJ	WLN	INT	LV	DCT	CALL	INT	LV
pc	0	0	0	0	0	0	0	0
q	41	0	4	4	5	0	5	4

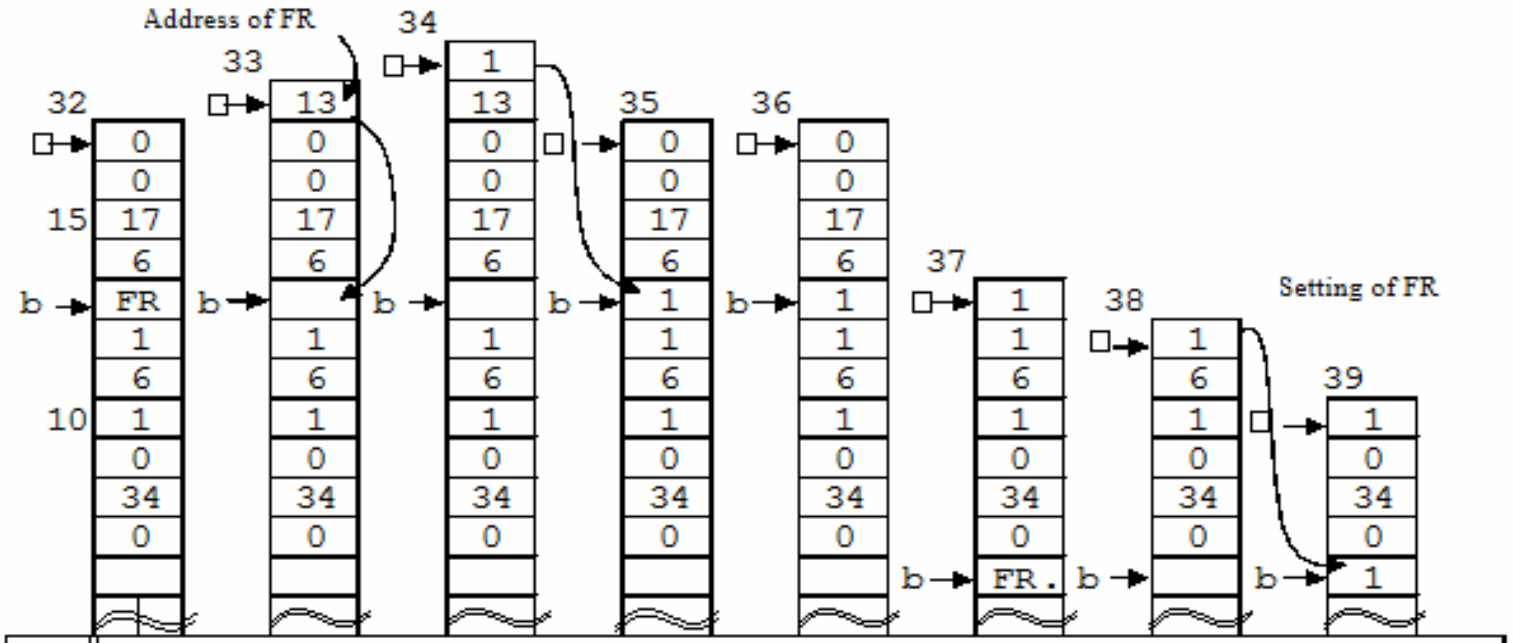


t	11	12	11	10	11	12	16	17
b	6	6	6	6	6	6	6	6
pc	2	3	4	9	10	11	12	13
op	LC	EQ	FJ	LA	LV	INT	LV	LC
pc	0	0	0	0	0	0	0	0
q	0	0	9	0	4	4	4	1

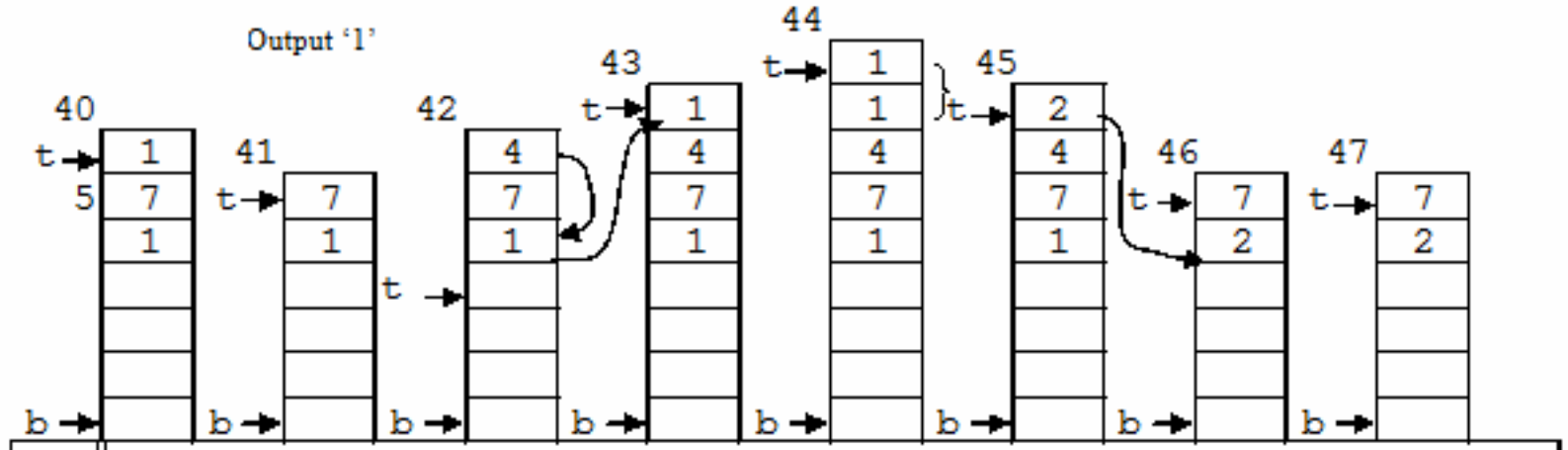


t	18	17	12	12	17	18	19	18
b	6	6	6	13	13	13	13	13
pc	14	15	16	0	1	2	3	4

op	SB	DCT	CALI	NT	LV	LC	EQ	FJ
pc	0	0	1	0	0	0	0	0
q	0	5	0	5	4	0	0	9



t	17	18	19	17	17	13	12	10
b	13	13	13	13	13	6	6	6
pc	5	6	7	8	19	17	18	19
op	LA	LC	ST	J	EF	ML	ST	EF
pc	0	0	0	0	0	0	0	0
q	0	1	0	19	0	0	0	0



t	6	5	6	7	8	7	5	5
b	0	0	0	0	0	0	0	0
pc	34	35	36	37	38	39	40	25

op	WRI	LA	LV	LC	AD	ST	J	CV
pc	0	0	0	0	0	0	0	0
q	0	4	4	1	0	0	25	0