f. `if a<b then c:=a else c:=b`, translated in a less clumsy way.

7.2 Translate each of these expressions into IR trees, but using the `Ex`, `Nx`, and `Cx` constructors as appropriate. In each case, just draw pictures of the trees: an `Ex` tree will be a Tree `exp`, an `Nx` tree will be a Tree `stm`, and a `Cx` tree will be a `stm` with holes labeled *true* and *false* into which labels can later be placed.

a. `a+5`

b. `output := concat(output,s)`, as it appears on line 8 of Program 6.3. The `concat` function is part of the standard library (see page 525), and for purposes of computing its static link, assume it is at the same level of nesting as the `prettyprint` function.

c. `b[i+1]:=0`

d. `(c:=a+1; c*c)`

e. `while a>0 do a := a-1`

f. `a<b` moves a 1 or 0 into some newly defined temporary, and whose right-hand side is the temporary.

g. `if a then b else c`, where `a` is an integer variable (true if $\neq 0$).

h. `a := x+y`

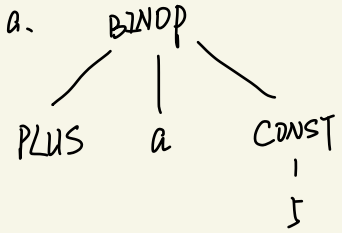
i. `if a<b then a else b`

j. `if a<b then c:=a else c:=b`

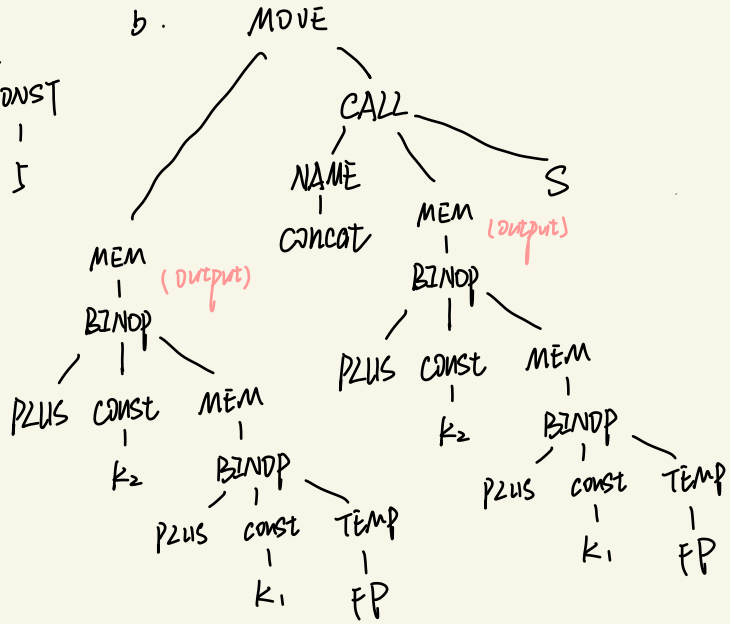
7.3 Using the C compiler of your choice (or a compiler for another language)

一般變量皆不使用 temp 對待

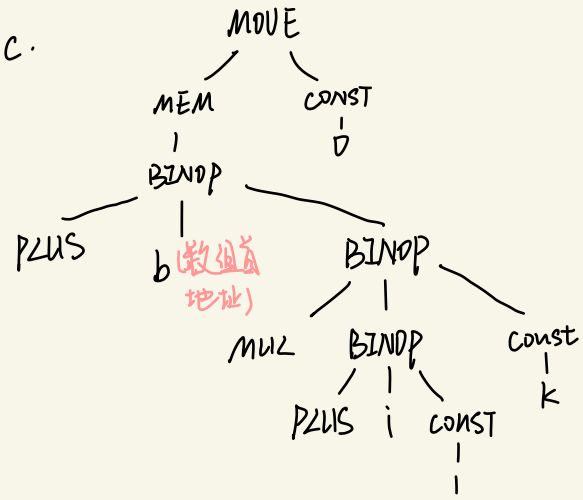
a.



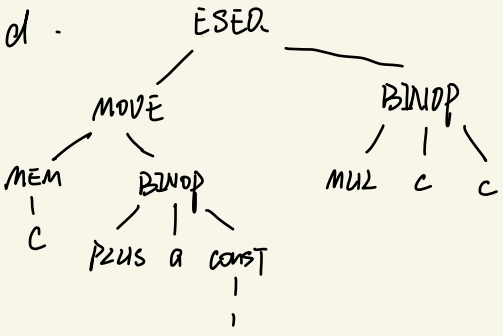
b.



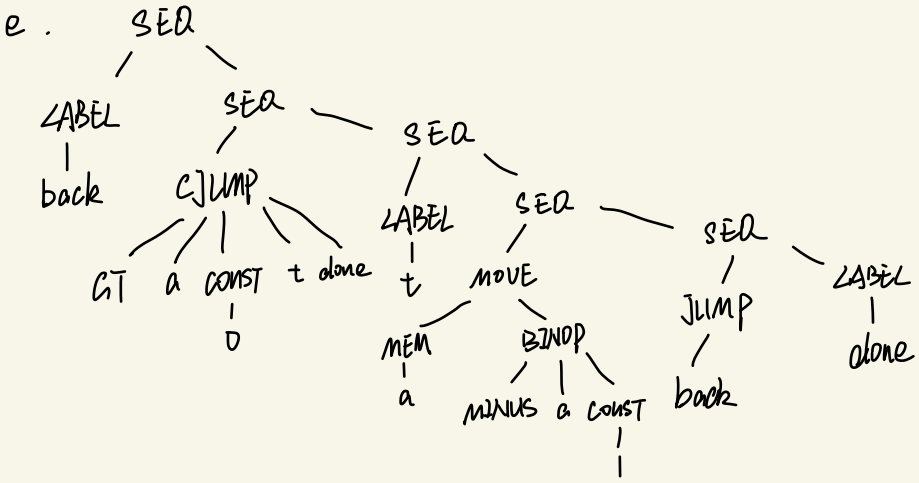
c.


MOVE
MEM
CONST
0
BINOP
PLUS
b
(数组首地址)
BINOP
MUL
BINOP
CONST
PLUS
i
CONST
1

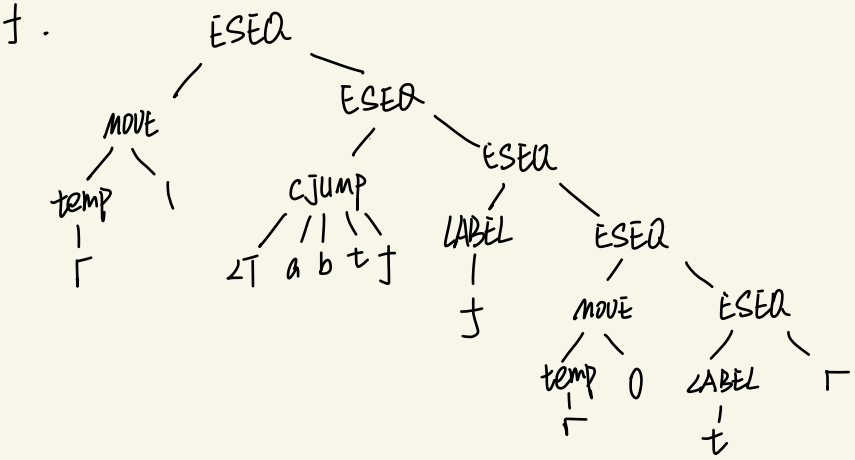
d.


ESEQ
MOVE
BINOP
MEM
BINOP
MUL
c
c
C
PLUS
a
CONST
1

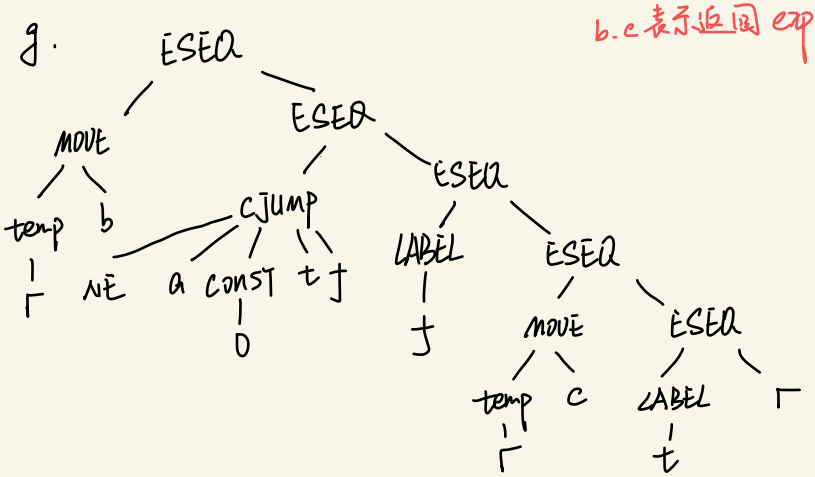
e.


SEQ
LABEL
back
SEQ
SEQ
CJUMP
LABEL
SEQ
SEQ
LABEL
GT
a
CONST
0
t
done
t
MOVE
JUMP
done
MEM
a
BINOP
back
MINUS
a
CONST
1

f.

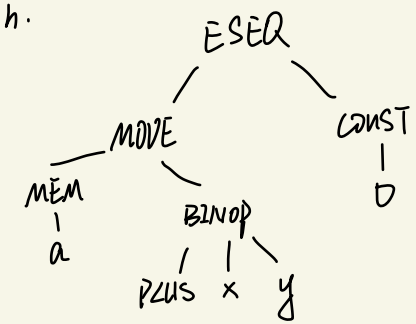

ESEQ
ESEQ
MOVE
ESEQ
temp
r
1
CJUMP
LABEL
ESEQ
LT
a
b
t
f
f
MOVE
ESEQ
temp
r
0
LABEL
r
t

g.



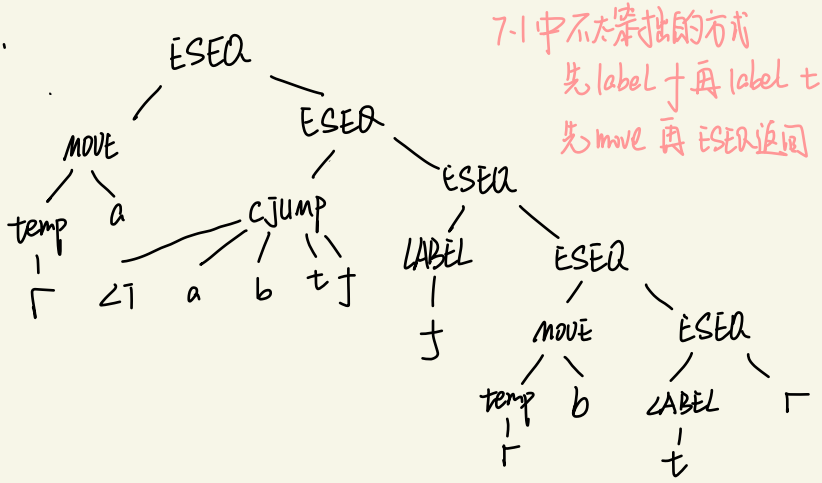
b.c表示返回eax.

h.



含返回EAX结点...要求(根据7.1推测)

i.

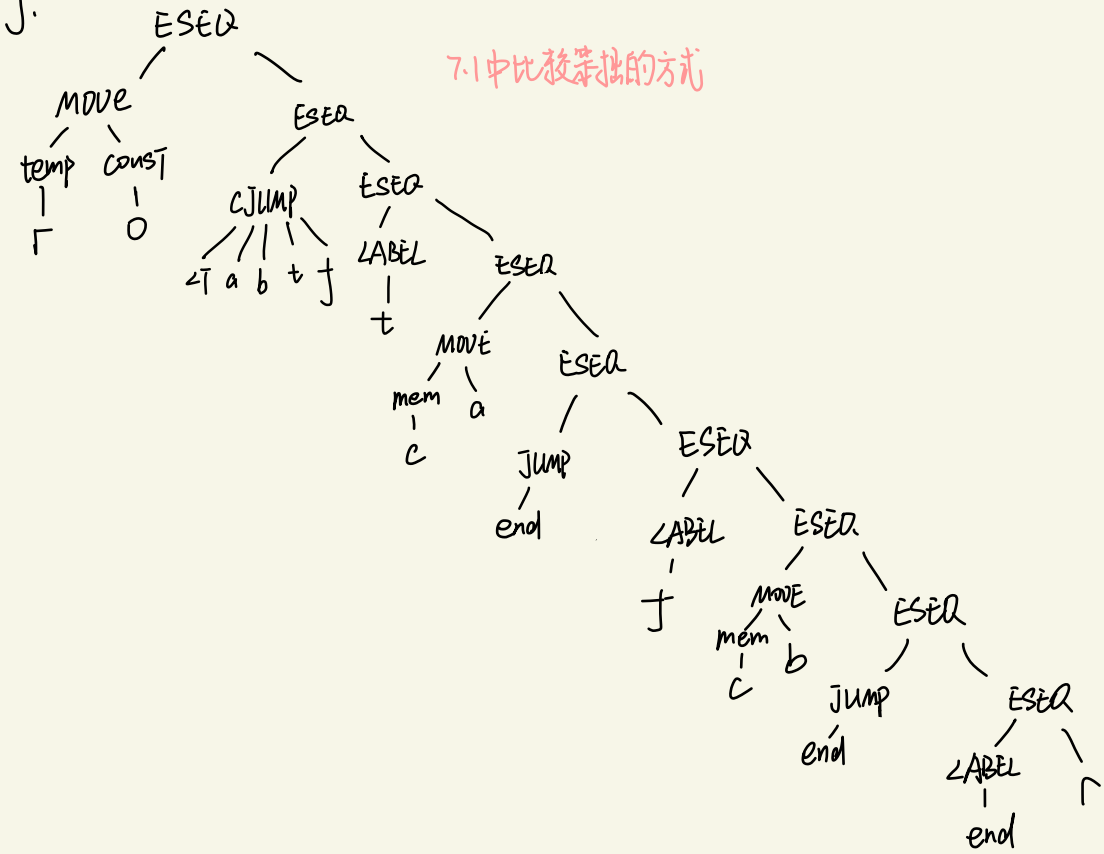


7.1中不太常规的方式

先label f再label t

先move再ESEQ返回

j.



7.1中比较常规的方式