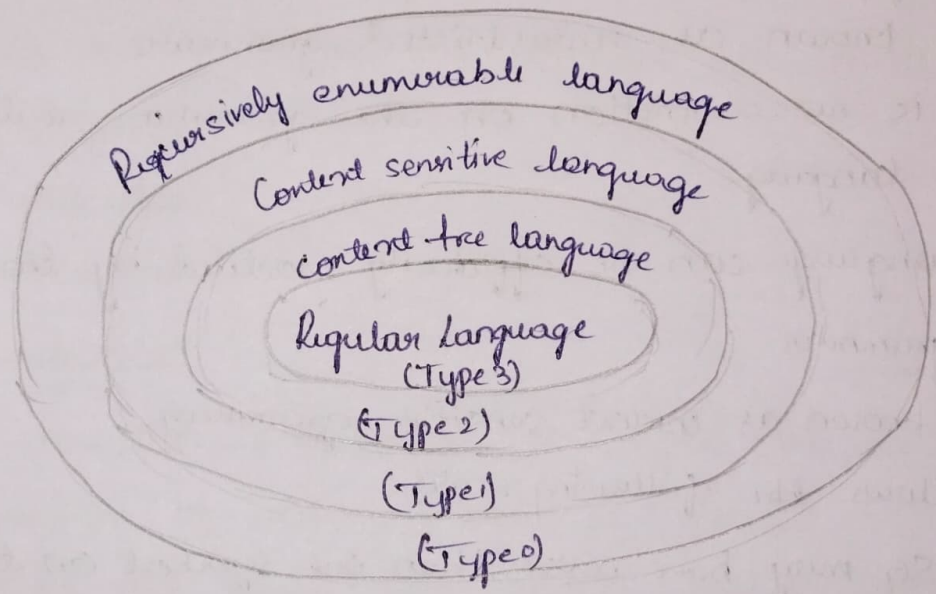
K. Manasa
21BS1A3326

ACD Assignment

1. Discuss chomsky hierarchy of languages and recognizers.

chomsky hierarchy of languages

(i) Regular language (Type 3)
(ii) Context free language (Type 2)
(iii) Context sensitive language (Type 1)
(iv) Recursively enumerable language (Type 0)



Chomsky hierarchy of lang represents classes of language that are accepted by different machines therfore every lang of type 3 is also of type 2, type 1 and type 0
Similarly type 2 is also of type 1 and type 0 & type 1 is also of type 0

Grammar	Language	Recognizing automata	Production rules	Examples
Type 3 regular grammar	Regular	FSA	$A \to a$ / $A \to aB$	$L = \{a^n / n > 0\}$
Type 2 (CFG)	Context free	PDA (push down automata)	$A \to \alpha$	$L = \{a^n b^n / n > 0\}$

Type-1 (CSG)	Context sensitive	LBA (Linear bounded auto-mata)	$\alpha A\beta \rightarrow \alpha\gamma\beta$	$L = \{a^n b^n c^n / n>0\}$
Type 0 (unrestricted grammer)	Recursively enumurable	TM (Turing machine)	$\alpha \rightarrow \beta$	$L = \{w / w\}$

Type 0 grammar

→ It is known as unrestricted grammar

→ There is no restriction on the grammar rules of this type of language

→ This language can be efficiently modeled by turing machine(T

Type 1 grammar

→ It is known as context sensitive grammar

→ It follows the following rules

(i) The CSG may have more than one symbol on the LHS of product rules

(ii) The no of symbols on LHS must not exceed the no of symbols on the RHS

(iii) $A \rightarrow \epsilon$ is not alone unless A is start symbol.

Type 2 grammar:

→ It is known as CFG

→ CFG lang are those language which can be represented by "CFG"

$G = \{V, T, P, S\}$ V-finite set of non terminals, T is finite set of terminals, P is set of product rules, S is start symbol of grammar (non-terminal)

Type 3 grammar

→ It is known as regular grammar regular language are those language which can be described using reg expressions

→ These language can be modelled by DFA and NFA

2. Explain different ~~stages~~ storage allocation strategies.

Storage allocation strategies

There are maily 3 types of storage allocations strategies

(i) Static Allocation

(ii) Heap Allocation

(iii) Stack Allocation

Static Allocation

- Memory allocation is determined at compile time and, the size and structure of memory needed are fixed.
- C and C++ use static allocation the memory will be allowed in static allocation

Advantages

1) Easy to understand

2) The memory is allocated once only at compile time and remains the same throughout prg compilation.

Disdavantage

1) Not highly scalable

2) Static storage allocation is not very efficient.

Heap allocation

It is used when the stack allocation lacks, if we want to retain the values of local var's after the activation record ends, LIFO schema doesn't work for the allocation & deallocation of activation record

Eg: C, C++, Python, Java

Adv

1) It is most flexible allocation schema
2) Heap allocation is useful where we have data whose size is not fixed and even change during runtime

Disadv

1) It is slower as compared to stack allocation
2) There is a chance of memory leaks.

Stack allocation

Stack allocation known as dynamic allocation Dynamic allocation means the allocation of memory at runtime
Stack is datastructure that follows LIFO principle.
So whenever there is multiple activator record created it will be pushed or popped in the stack as activation and ends.

Access is

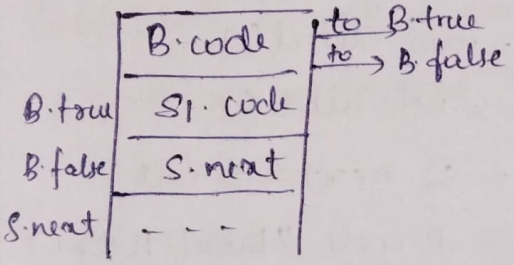
3. Explain about translation of control flow statements

Translation of simple and control flow statements

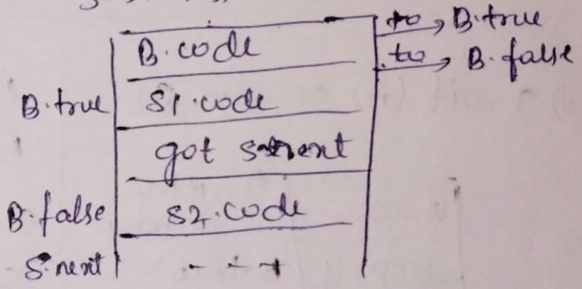
The translation of boolean expressions into 3 address in the context of statements such as

1) $S \rightarrow if(B)\ S1$
2) $S \rightarrow if(B)\ S1\ else\ S2$
3) $S \rightarrow while(B)\ S1$

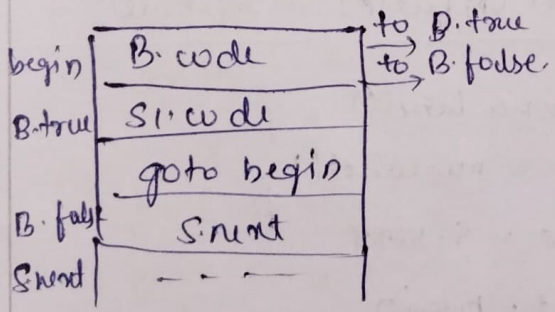
$S \rightarrow if(B)\ S1$



$S \rightarrow if(B)\ S1\ else\ S2$



$S \rightarrow while(B)\ S1$

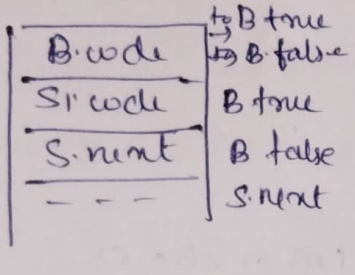


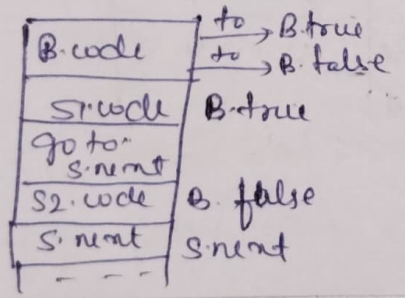
with a boolean expression B, we associate 2 labels

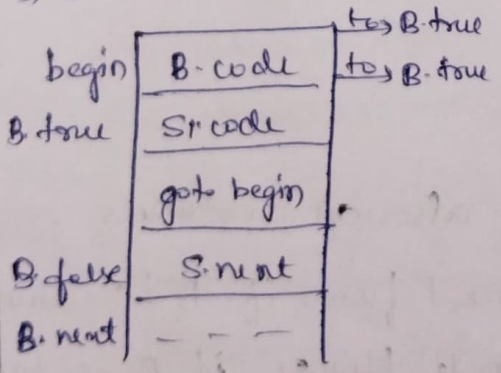
1) B.true - the label to which control flows if B is true
2) B.false - the label to which controls flows. if B is false

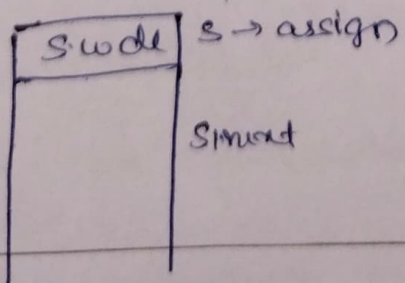
S.next denoting a label for the instruction immediately after. code for S

Statement and semantic rule

Statement	Semantic rule
1) S → if (B) 	B.true = new label() B.false = S.next = S.next S code = B.code // label (B.code) // S_1.code.

2) S → if (B) S_1 else B_2 	B true = new label()
	B.false = new lable()
	S1.next = S2.next = S.next
	S.code = B.code // label(B.true)
	// S1.code // gen ('goto', S.next)
	// label (B false) // S2.code

3) S → while (B) S_1 	begin = newlabel()
	B.true = newlabel()
	B false = S.next
	S1.next = begin
	S.code = label (begin) // B.code.
	// label(B.true) // S.code
	// gen ('goto' begin)

4) P → S 	S.next = newlabel()
	p.code = S.code // label (S.next)
	s.code = assign.code

5) S → $S_1 S_2$	S.code = S1.code // S2.code.

4. Write the characteristics of peephole optimization

Peephole optimization is a technique for locally improving the target code which is done by examining a sliding window of target instructions and replacing the instruction sequences within the peephole by shorter or faster sequences wherever possible.

Some characteristics of peephole optimization.

1) Redundant instruction elimination

- Eliminatory redundant load and store

```
LD R0, a
ST a, R0
```

→ eliminate either load or store to eliminate redundancy

- Eliminating unreachable code

```
if dbg == 1 goto L1
    goto L2 (unreachable code) *
L1 print debugging info
L2 print unreachable code
```

2) Flow of control optimization

Here unnecessary jumps can be avoided

```
goto L1          goto L2
L1 : goto L2     L2 : a = b
L2 : a = b
```

(unnecessary code)

3) Algebraic simplification

x = x + 0 → (unnecessary) x = x * 1

x = x * 1

both results same x = x + 0 is unnecessary algebraic expression

4) Use of machine idioms

i = i++

i = i-1

Two operation difines increment and decrement operations which can be replaced by equivalent machine instructions INC & DEC

INC i

DEC i

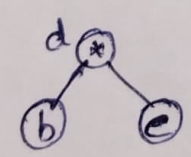
Peephole optimization objectives are:

1) Improve performance
2) Reduce memory footprint
3) Reduce code size

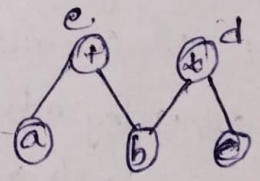
5) Construct DAG for following basic block

1) d := b*e
2) e := a+b
3) b := b*c
4) a := e-d

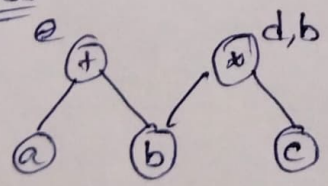
Step 1: d := b*e



Step 2:



S3:



S4: