Code Optimization
Cade ophimization is an important part of compiler design. It is highly integrated in the cade generation phase. This is because in partant aspect of cade ophimization is the efficient use of registers and text can
only be performed during code generation.  It is depressed recognizing patterns & substituting team for efficient on  Code optimization has two basic grals'-
1) The ophnized code Should be more officions then the oxiginal code in terms of signate and for sine neguirous
2) The oplinized cool should perform ter exact some functions as the ariginal code. This mains text a correct
incorrect on while trying to optimize
Other tem to objicient use of registers, andler very important part of cacle appinization is loop ophinization. This is because 86%.  It him is a program is spent in 20% of the code -> that is in the book.  Thus if her ophinize the code in a loop the whole program works more efficiently.
Main types of code aptimization:
2) Common sub expression removal 3) Loop ophinization.
Scanned by CamScanner

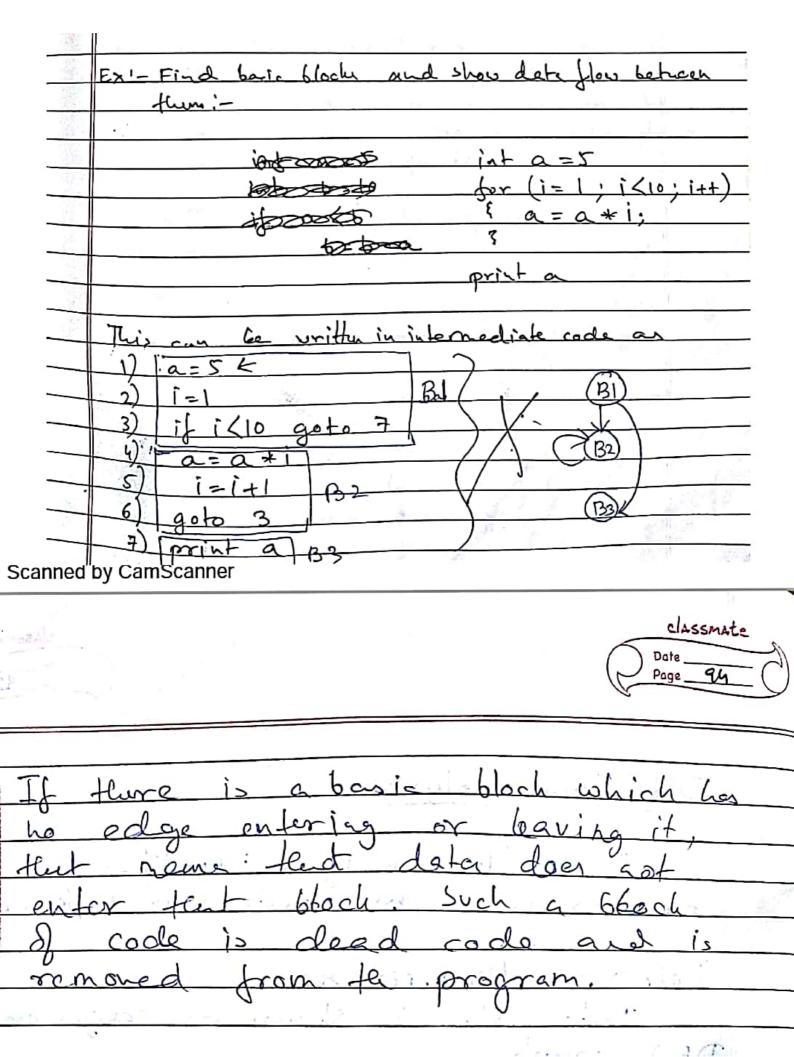
	Common Subenpression Removed 1000
	For a place of code where values of variables
	do not change, compiler may use a
	temporary nariable to same a common cuboxpression.
	and neuse it as many times as needed.
	$Ex \rightarrow a = (b+c)' - (d+e) + (b+c)$
	f = (6+c) - d
	to be see and and the
	applinized coder animal and Co
	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
	$t_1 = 6 + c \times 10^{-1} \times $
	$a = t_1 - (d+e) + t_1$ $f - f - d$
	t = t, -de la manufación de la manufació
	* This can only be done if the value of 6 &s
(RO)	don noch dange in between *
	militarial service ser
	0 1 0 0 1 /10 - 1 11 c-do'-
4305	Removal of Dead (Unroschable Code -
200	A piece of code that cannot be reached
	by under any circumstance is called dead
1	cade or unreachable code. The code ophimizar
Visite S	can identify such blocks of code using data
1 State	flow analysis and opiningle tem.
- Grand	
	eg) if $(a = b)$ print a
• • • • • • • • • • • • • • • • • • • •	aln il (a < b)
30%	print and
-	else if (a>6)
t	print 6 surreachable code.
	else print 6
Scanned by	CamScanner
saioa by	

Loop Ophinization:  Leap aphinization is a very important aspect of code aphinization. In care of nested (cops, if makes a large difference in  Leap inversant  2 Loop jaminization involves!  1) Loop jaminization involves!  2) Loop jaminization  3) Loop various  3) Loop various  4 loop jaminization  1) Loop jaminization  1) Loop jaminization  2) Loop jaminization  4 loop jaminization  4 loop jaminization  4 loop jaminization  4 loop jaminization  A code appinization  Loop javariant  A code appinization  Loop javariant  Lo
Loop aphinization is a very important  aspect of mode appinization. In case of  mested (cops, if modes a large difference in  the officiency of the prooperson.  Loop invariants  2) Loop Janning  3) Loop Janning  4) induction variables.  Invariants  Doop juvariables—  Some variables—  Loop juvariables—  Loop juvariables—  Loop juvariables—  Loop juvariables—  Loop juvariables—  Loop juvariables—  Loop juvariants  A Code appinaizer places  Loop juvariants—  Loop juvariants—  A Code appinaizer places  Loop juvariants—  Loop juvaria
moded (copy if maker large are usell appinized, if nakes a huge difference in  the Officiency of the program.  Large ophimization involves!  1) Loop jarming 3) Large unralling 4) induction variables.  1) Loop jarming  Some variables.  Loop jarming  Loop
moded (copy if maker large are usell appinized, if nakes a huge difference in  the Officiency of the program.  Large ophimization involves!  1) Loop jarming 3) Large unralling 4) induction variables.  1) Loop jarming  Some variables.  Loop jarming  Loop
moded (copy if males a large difference in  the Officiency of the program.  Large ophimization involves!  1) Loop jarming  3) Large unvalling  4) induction variables.  Some variables do not charge facing value withing a large but any get and prodefined everythan to loop invariant.  A code appropriate on a large run. Such elements are called loop invariant.  A code appropriate places loop invariant outside to loop:  Cod for 121 to 10  S reverso (a=1) > loop invariant.
Lang aphinization involves!  1) Loop invariants  2) Loop Janming  3) Loop unvalling  4) induction variables.  1) Loop juvariables— Come variables do not charge  facir value within a loop  but may get and redefind  everythere to keep is being  run—Such elements are  called loop invariant—  A Code aphinizer places  Loop lavariant outside te  loop:  cod for let to lo  s reverso a= 1-3 loop invariant  moreone b= it.
Lang aphinization involves!  1) Loop invariants  2) Loop Janming  3) Loop unvalling  4) induction variables.  1) Loop juvariables— Come variables do not charge  facir value within a loop  but may get and redefind  everythere to keep is being  run—Such elements are  called loop invariant—  A Code aphinizer places  Loop lavariant outside te  loop:  cod for let to lo  s reverso a= 1-3 loop invariant  moreone b= it.
Lang aphinization involves!  1) Loop invariants  2) Loop Janming  3) Loop unvalling  4) induction variables.  1) Loop juvariables— Come variables do not charge  facir value within a loop  but may get and redefind  everythere to keep is being  run—Such elements are  called loop invariant—  A Code aphinizer places  Loop lavariant outside te  loop:  cod for let to lo  s reverso a= 1-3 loop invariant  moreone b= it.
Lang aphinization involves!  1) Loop invariants  2) Loop Janming  3) Loop untalling  4) induction variables.  1) Loop juvariables— Come variables do not charge  facir value within a loop  but any get auf redefind  everythere te (sop) > being  run—Such elements are  called (cop) invariant—  A Code reptionizer places  Loop lavariant outside fee  (cop)  2 presence a= 1-> loop invariant  moreone b= it.
1) Loop janning 3) Loop janning 3) Loop janning 4) induction variables.  1) Loop januariables— Come variables do not charge fecir value within a loop but any get and reclepted everythere the loop is being run—Such elements are called loop invariants— A cacle replimizer places loop lavariant outside te (eop:  ca) for i=1 to 10  5 remains a=1) > loop invariant moreous b=i+1
2) Loop janming 3) Loop unvalling 4) induction variables.  1) Loop javariables— Come variables do not change their value within a loop but may get and redefined everythme the loop is being run. Such elements are called loop invariant.  A Code reprinciper places Loop lavariant outside te loop:  2) for 1=1 to 10 2 tomorrow a=1) > loop invariant morrows b=it.
Deop invariants - Com variables do not change  their value within a loop  but any get and modefind  energial to loop invariant.  A code optimizer places  loop lavariant outide to  (oop:  call to 10  S romans (a=1) > loop invariant  moreons b = it!
Deop invariants - Com variables do not change  teair value within a loop  but any get and redefined  everytime to (sop is being  run: Such elements are  called loop invariant.  A code optimizer places  loop lavariant outside to  loop:  cod for i=1 to 10  s romans b=it!  c=b+a
Deop invariants - Com variables do not change  their value within a loop  but any get and modefind  energial to loop invariant.  A code optimizer places  loop lavariant outide to  (oop:  call to 10  S romans (a=1) > loop invariant  moreons b = it!
Deop invariants - Com variables do not change  texis value within a loop  but any get and redefined  everytime to loop is being  run: Such elements are  called loop invariant.  A code optimizer places  loop lavariant outside to  loop:  called loop invariant
teric value within a loop  but may get and modefind  everytian the loop is being  run. Such elements are  called loop invariant.  A code rephinizer places  loop:  co) for 1=1 to 10  S remain a=1) > loop invariant  run a=1+1.
but may get and modeficial  everytime the (bop is being  run - Such elements are  called loop invariants -  A code optimizer places  loop lavariants outside te  loop:  20 for ist to 10  8 november (a=1) > loop invariant  morrown b=i+1.
every has the loop is being  run - Such elements are  called loop invariants.  A code appinizer places  loop lavariants outside for  loop:  co) for i=1 to 10  someone b=i+1.
every has the loop is being  run - Such elements are  called loop invariants.  A code appinizer places  loop lavariants outside for  loop:  co) for i=1 to 10  someone b=i+1.
Callad loop invariant.  A Codo reprinizer places  Loop lavariant outside for  loop:  Callad loop invariant  outside for  Loop lavariant  someon (a=1) > loop invariant  remain b=it!
A code aphinizer places  loop lavariant outside te  loop:  cod for i=1 to 10  s nomen a=1 > loop inversant  nomen b=i+1.
(oop:  (oop:  Sor i=1 to 10  Someon (a=1) > loop inveriant  someon (b=i+1)  (=b+a)
(oop:  (oop:  Sor i=1 to 10  Someon (a=1) > loop inveriant  someon (b=i+1)  (=b+a
co) for $i=1$ to $10$ S nomes $a=1$ $\rightarrow 10$ inversant  Somes $b=i+1$ $c=b+a$
Sor i=1 to 10  Someon a=1 -> loop inveriant  Monoson b=i+1.
Some $a=1 \rightarrow loop inverious$ Some $b=i+1$ $c=b+a$
3 C=b+a
3 C=b+a
3
a does not change it werke in the love
and hance may be blaced outside the life
to enhance performance.
canned by CamScanner
DateC)
Page _99_
0 = 1
for i=1 ho 10
7 - 5-1
2 = 6+a

	and the labor to art
2)	Loop Jamming - of ten, programmer, for ter
	sale of includ improvis
1 1 2	school by use separate
	text can be performed in
	a single loop. The coole
- 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	optinizer identifies & Suses
	such loops.
	ea) for 1=1 to 10
	A(1) = 0 ;
	for jal to 10
	B(j)=0; d(2) 1 1 1 2
	dir (186 mills - work)
OF	shinized code: - I am and
V The state of the	
	for ist to 10 mil thin
1986	\$
	A [i] =0
	3
1	
Scanned by	CamScanner
i	
3/	can Invalling !- Canabia it is more affectable
	· · · · · · · · · · · · · · · · · · ·
-	
\	loops are eneruted. This is
	because it somes the contra
\	Step & updat - to Coop courter.
<u>}</u>	This is alled loop enoulling
	50) 1. ( ) .
	5 (1=1; hx10; 1++) - 3 (3x10=20) 22 show
1	
	3
	print sum -1
1	
1	
3) 1	sop Unvolling! - Some bing, it is more efficient  by soci increase for number of  otherwork in a loop them  and decrease to number of some  loops are enecuted. This is  because it squess the contra  slep of updating to loop counter.  This is called loop annothing  son=0  Eg) for (i=1; ix10; i++) 2  Sum = sum+1 1  Sum = sum+1 1

applinized code:	
,	
Sum = D	
for (i=1; i<10; i=i+2)-27	
3 \4x5 =20\2	3 sh
Sum = sum +i 7b	
sum = sum + 1+1	
3	
print sum -1	
Thus code runs 10 steps less in	Se chi
Social sinks to sign was the	
Case.	
nea by Camscanner	
Oate al	)
4	
Winduction Variables! - Some calculations in 60	OP.
The director mes	1 hed
and brought outsi	,
like cop invariants	<u></u>
colly some other variable	
outside te cop into pur	P 0 24
can be fulfilled. such	ably
variable which indice	The
ophnige a loop are to	<i>-</i>
variables.	-
Designation of the state of the	
Fa) Form I=+	
tox 1= 1 b 10	
2 A[i] = T* 4 - 4	
] I=I+2	
i bearing state much some in little in some	
the statement A[i] = 7x4-4	
is broken into laternatiate code as	
The I + 47 h off to	
72 = Ti-4.	
and bridihops a Ali]=It Inalde (8	
and the second of the second o	414
IF justiend are ductore I from O, our jo	roger
becomes:	
J=0 - which along the first	
for i=1 to 10 , internadiale code	1,
12 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	
Cold Fill A (II) = I+4) A (II) = Fill A	
$\frac{A[i] = I + 4}{I = I + 2} A[i] = I_1 + 4$	
A (i) = I+4) > A (i) = A, b, as	

-		O Dola Flour Andrewish
1		Basic Blacks & Data Flow Analysis:
\	_	An important aspect of code ophinischin is to roll his cohon of bosic blocks.
-		is the educification of Dayle Blocks.
-		A basic block is a block of code which
		start wife ter first statement and
<u> </u>		seguntially enecules statement one after
		A basic black is a block of code colich start with the first statement and sequentially executes statement one after the other without branching
		Each basic black begins with a loader stalement.
-		statement.
-		
_}_		Identifying leader statements:
		Loader State ment have to following parters.
_\_		1) First statement of a program is a loader 2) The target of any conditional or unconditional jump is a leader 3) statement pollowing a conditional jump is
1		2) The target of any conditional or
		unconditional jump is a leader.
-		a Gader.
<b>U</b> —		a Gadely.
		Identifying basic blocks:-
		A basic black begins with a bader and ends at the last statement offens
		and ends at the last statement offine
		the nent bader.
44		
144		
Scann	ed by	CamScanner
		Date Page 43
_		
	_	Data Hessi analysis
		Data flow analysis identifier how data
		travels from one basic block to ter rent.
		Data & la con des foras mas
		Data Stew is shown as edges on the
	7	FD Graph.
_		There is an edge between two basis
	1	tocks if and only if:
		is a jump which are to the ladger
	San	



	Cade Generation
	Control of the second s
	Cade generation place of the compiler takes the optimized intermediate code on ipput and tries to generate farget code for ten target machines
	takes the optimized intermediate code on
1	ipput and tries to generate farget code
	for ten target machine
	The input consist of 3-address code,
	postfix not about and syntax trees or DAGE
	It takes There lapots and tries to generate
	The input consist of 3-address code postfix not a hour and synten trees or DAG.  It takes These upputs and true to generate an efficient target code.
	The code generator may make several intermodiate codes and pan term through the code optimizer before making the final target code
	intermediate cooler and para term through
	the code optimizer before meeting the final
	target code
Control 1	
	the final code generated must have
	the final code generated must have be following properties:
The second second second second	
75	or the source program and should be of high qualify
	he of the culif
	The state of the s
1 1 1 1 1	oft should make only
The state of	The should make optimum use of the mesources of the tanget machine.
-084	19 18,500 cas of 12 ranger mechice.
A THE	denor and seed add or a seed of the
- 20.00	Code optimization has It &
-	Code generation involves the following actions:
	TAMES OF THE PROPERTY OF THE P
	1) Jenstruction Salection
. Uhran	2) Register allocation
	3) Instruction ordering
La tentre	Contraction of the state of the
canned by	CamScanner
6	dessnate
	Page 96
Å	
	Instruction Selection - This involves finding for
Tt.	Instruction de appropriate machice
	language instruction to
14-	implement each like
114	ta luter mediate codo
1	
<u> </u>	Register Allecation - This involves predicting
	Which registers are qualalle
H	and what to store in
H	which register
	This involves desiding te
	Instruction Ordering - This invalues desiding to
	instructions are to loo
	7-0-11
4	executed.

Output of code galerini
I and senerator produces target
The code generators produces target code which can be of 2 types:
i) RISC
L) C15C
The second of th
1) RISC - reduced just ruchion sat computers
This type of machine how than registers,  many 3 byte a didnesses mands, simple addresses  modes and simple instructions.  Complem
2 loute a aldressale and in the state of the
and so of six of
Condian
2) (154 - ( - ++++++++++++++++++++++++++++++++
2) CISC - Complete instruction set computerio
This Las O is a
2 - Lile address has few registers
7-byte addrenies medos, namy addressing moderning register classes a lineary instructions.
Many register closes and many instructions.
by CamScanner
Classmate 3
Date 97
No.
Register Allocation - The code generator has
to decide upon the matter
allocation text is to be dans
for each Instruction in the
program - Ragister allocation
eaun in too stepi-
Desidir calife warlable needs
Deciding which point is hime
attorias and assessment of the second
2) to 1: white maisles are
2) Finding what register and available and allowing them
accordigly.
Instruction Used by Code Generator:
A 12 (12 )
I Load operation - Loads value into specific
negister III
The same of the sa
1D (deshinahim, source)
2 2 2 2004
2 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2) Computational operation - Performs operation specified such as
specifica such as
ADD, CUB, MUL, DIV
100 mg 1 m
OP (distination, vource 1, source 2)
Or australian, hourse 1, source =
数と Sunfaction Sun and Sun an
Alexander and the second secon
3) Branch statements!
21 Dranch Statering
1) Unconditional Jump >
BR (L) - Jumps to label L.
IDIN JUZ. Jail

3) Branch statements !-	
1) Unconditional Jump >	_
BR (L) -> Jumps to label L.	
1512 \LZ. Jan.	
	وني در
CamScanner	
The state of the s	1
classmate	1
Date	)
2) Carditional Jump ->	_
2) Concurrance Course	
DC 1 - checks value in	
scond r, conster r, if	h
250 jumps to	
Une L.	7.
	2
Ex- write target nachine code for following jutomediate code!	
Exi- Write target Addice	
(whormediane coll	_
$T_1 = 10 * 6$	
T - T / C	
$T_2 = T_1 + C$ $d = T_2$	
Q 12	
7.0 2. 0	
LD R <sub>1</sub> , a	
MUL R2, R2, R2	
ADDO BADBET	
10 Rg, c	
$ADD R_2, R_2, R_3$	_
ST R2,d	
Many political operation and the safety	—

	Exror Handling
	Luch with all
	Error handling routines interact with all
, S	Stages of compilation. Whenever the compiler
	faces an error the error haraller tries
	to repair the order of according
	a) compilation In can be errors, a
	compiler is not expected.
	i) Stop entire process of compilation at first detected error.
	1) Stop entre process of the
ja en	pret deregen and
	2) Generate a woong output
1	
<u> </u>	3) Initiate a system crash.
- 10	· midde - median +
1	A good compiler should try to at least
	the property so the sure
	car be defected.
	A more advance ampiler may try to repair the error using a number of
	A more advances compiler may ry
	repair the error using a number of
	strategies.
	Whenever an error is detected, the
	Compiler should generale an error
	mers age which will be displayed
	at the end of the compilation process
	at the end of the compilation practice
annea b	y CamScanner
	catures of a good error diagnosis=
	to a supplied of the supplied
	1) The error merrage should pinpoint the
	remor in terms of source code and
	not in terms of some internal
	representation.
	2) The array 1 11 by L. 12011
	2) The error neesage should be fasteful and cleanly defined
	The state of the s
	3) The error merrage should state ter
	error understantably and not in
慢	terror of some obscure error code
	gets to topped in it is against to
150	4) Foror messages should not be
37	redundant.
	believes in the war experience and the fine of
	considerate distributions interes management and

Lexical Phase Errors :-
herical phase error occur when an
invalid beyword is present or an ignalid
lawical syntan is generated benical errors
are pary to bocale borrows The lexical
analyzer may perform on of two achians
when fored with an error:
- the start with more and the start of the s
DIt may ship the roughied token and
search for the went levene This
generales a deleted character error
The state of the s
12) It may try to repair the error by
minimum distance matching or
string matching techniques so as to
avoid further errors.
Scanned by CamScanner
Date C

Scanned by Car	mScanner
	DoleOB
	Minimum Distance Matching:  For an erroneous input 'a' in the lonical analysis phan, the analyzer tries to repair it by performing one of the following repair it by performing one of the following repair it by performing on existing valid error transformations on existing valid keywords so as to arrive at in
	Error fransformations include:
	1) Inserting a character  2) Deleting a character  3) Transposing two adjacent characters  4) Madifying a character.  Lenical phese errors can be corrected  by various string pratching techniques.
	Strings: Parts of a string:-  A string can have the following parts:
	Profin > remard of 0 or more letters from the end.  Suffin > remard of 0 or more letters from the short  Sub string > removed of a profin or softin or poth.
	Proposer Prefix & Suffixer > All prefixer & suffixer except te striy itself  Subsequence > Deletion of one or more letters from strip

FJ III	
	Suffix & renoval of O ar more letters from the short
. 14	
	Sub string - removed of a profin or softin or foth.
	also and the state of the state
	Proper Prefix & Suffixer > All prefixer & suffixer
	Propoer Prefix & Suffixer -> All prefixer & suffixer except te itriy itself
	and continue of the first the second
	Subsequence & Deletion of one or more letters from string
	The second of th
Scanned by Car	nScanner

u L	by CamScanner
	Date
	Ex! - Generale all prefiner, suffiner, substriegs, and Subsequences of the word
	THE REPORT OF A SHOULD BE ASSETTED AS A SHOULD BE ASSETTED.
	HERMIONE AND MAN ALL
	Profix > H, HE, HER, HERM, HERMI, HERMIO,
141	HERMION, HERMIONE
	The state of the s
	Suffix - HERMIONE, ERMIONE, RMIONE, MLONE,
	TONE, ONE, NE E
W.	
100	Substring > H, F, R, M, I, O, N
8	HE, ER, RM, MI, 10, ON, NE
<u> </u>	HER, ERM, ANI, MIO, ION, ONE
-205	HERM ERMI RHID MICH . IONE
	HERMI, ERMIO, RHION MIGNE
3).	HERMIO, ERHION, RHIONE
4	HERMION , FRMIONE
	HERMIONE
16	Subsequence > H, E, R, M, I, O, N
202	HE, HR, HM, HI, HO, HN
12	ER, EM, EI, EO, EM, EE
	RM, RI, RO, RN, RE
	MI, MO, MN, ME
·题	10, IN, (E
4	6N, DE
	NE
	HER HEM, HEI, HEO, HEN, HEE
1	· Lababala
J E	
	who will be a little of the li
	Frot washing purther time +
	y CamScanner

and the same of th
Syntarchic Errors!
Syntactic errors are the casiest to
spot for the process
syntan analysis follows strict roles  Syntan analysis follows strict roles  By content free gramman to generate
of content free grama on
parse free.
Whenever ten parser finds itself
in a position in the parce tree from
in a past was as least move left, it
which it has no legal move left, it
knows it has detected an error.
The sparse tree decides it has faced
an error by judging it state, stack
contents and input string.
Taladla le arma d'al la 11
Ideally, the parier should be able
to detact the error encountered and
recover from it to continue parsing
the most of the opposes Input.
the second of the classical base of
and a third that the same
Panic Made: - A parser gass it
Panic Mode: - A parser goes into panic mode
when it cannot meconsigue
itself to resume parsing
h panic made, te purser
reject all input till it hids
a synchonanizing token such
an a dali il tolen such
as a delimiter (In or;)
It then empties it stack till
Synchronizing boling and
ten resumes paring The
advantage of this method is
that is this method is
that it is easy to implement
imScanner

Storage Allocations
Achim Lee and the Achim Lee an
The allowaling a
The allocation and deallocation of data
packages which call subroubies to hangle
allocation and Deallocation of data object.
Achivation Traces:
ahoch shows the southern Dinter
are te da !
recursive procedures
Point in connection to activation tree;
since the achien tree!
· Each nade represent an active mande
ah achee procedure
. The root represents the mails
2001 represents the main Josegram
· hade 'à is ter parail 1
nøde 'à achimles made it
Andrew Commenced to the second control of th
nerde a Don te lett a
node 'a's lifetie of mode bi
balone hade b's enecution ends
The state of the s
the state of the s
The state of the s
A CONTRACTOR OF THE PARTY OF TH
Scanned by CamScanner

	Activation records
	an achieria record this record is
N. L.	main tained by the run-tie stack. The
	achiebion verords in the stack show ter
	order in which the nodes In an
	tast achieved node has its record on
	dop.
	T
	(Ce information recorded in an activation
	The information recorded in an activation record way from language to language For PASCAI, activetia records containi-
the same	A Day of the control
	1) Temporaties -> All locally computed data. It
	1) Temporation > All locally computed data. It is not be actual registers, metaly the word variables
	the second secon
	2) Local Data > All Escally declared variables
- 1	and date
	3) Access Link + Contains links to all tendate
	Heat is being accessed by
	the record
	A) control link - Contain a pointer to be
	caller ractice
	(c) (a. 1.2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
HOT	5) Saved Marchice States > Contains reformation
	about te strate of le control strate before
14	Execution of subrastice
	When subrowh's ends, this
	value is reset into contralstee
canned by Ca	mScanner I
6) 5	pea for Return Value:> If the subrowhile roturn
	any volue, It is cared
	- cl and martha to a south of the section of the se
1	tokal Paraneters -> Their is when the data
	is acholy said in
	de registers II à
	negained for efficient
1 41 10	date storage.

	Stack Allocation: - In stack alboration, memory
41	(are how are assigned every
2/1/20	the a procedure à called
	All skakai members vaal by tu procedure is hept
	in flui nency location.
1	Ona le procedure D
	oner, tee stack automatically
	deallardes ter monony
	used by few procedure.
	Start allocation is especially
	useful for recursive procedures.
74	_0 0
	Fa)
	deallocated once ((2) i) over
	(())
1	800
4	main()
1	all re-
ied by CamS	Scanner
#	
AI AI	Idoro lo - 1
heap M	Bocchian - Another method for dynamic
	allecation is homp allecation.
1	Heap allocations allocate
	money spaces in configuous
	tocations and keep a link bother
- 14 m	then - when a namony space
	is deallocated it is liched
	to te available free space
W monday	deallocated list
WARD	180 2 180 4 1 - 1 (1) 3
	7 7 3 7
- 11	allocated list
man well on	The state of the s
calling of the	Deallocation of menory may be
- 1	amplicit (as in Pascal) or
19 1 11 121	implicit (as in USP)
2 17 Marca	The many location may be
YOU WITH THE	
	of find size or vertable size
3	

For blocks of variable size, the
morning of frequented over
fine with each block of free
manay being add to to to to
do deallo cated nervry.
Now allocation is done by
FIRST FIT needlad test is for
a block of code of size s,
tu first location of size f
such feet SS & Tribosen and
6 allocated to it leaving a
consecution blocks are free ten
are joined to form a larger blook.
Scanned by CamScanner
Storage allocation Structure of C:
and shock he of C.
C uses day on him l' A al l'
dynamic allocation techniques.
Memory is divided it for
Memory is divided into four pert:
THE PARK.
The comprises of all codes and global variables.
It comprises of all and transcolly.
variables.
Stack and Heap are dynamically stores
in some many location but fruits
in some money location but fooding opposite direction of growth
Achination records are stoned in
are stored in te heap.
Code & Data
3 Shakh
Johnen
4
Theip
Man
Memory Allocation in C.
A A A A A A A A A A A A A A A A A A A
The state of the s