CD-Assignment-2

Name: G. Rishisai
Rollno: 22481A0580
Class : III B tech CSE-B

1. Consider the following C code and generate it contents of Symbol table.

Sol: Definition of symbol table :- It is a data structure used to store the names of identifiers, scope and then binding information, it is a part of compiler which is accessed in all the places the attributes of symbol tables are

① name ② Type ③ size ④ Dimension ⑤ Line of Declaration ⑥ line of usage ⑦ Address

Example :-

① Void main() ② { ③ int a; ④ int arr[10];
⑤ int b, c; ⑥ a=10, b=20; ⑦ c=a+b ⑧ }

Symbol table :-

Name	Type	Size	Dimension	LoD	LoU	Address
a	int	2	0	3	6, 7	2040
arr	int	20	1	4	–	2096
b	int	2	0	5	6, 7	2000
c	int	2	0	5	7	1000

2. Explain live variable analysis with suitable analysis.

Sol: It is a specific technique that is implemented to optimize register space alocation for given place of code and facilitate the procedure for dead code elimination.

$$IN(n) = USE(n) \cup (OUT(n) - DEF(n))$$

$$OUT(n) = \cup IN(s)$$

$s \in$ successors(n)

Where

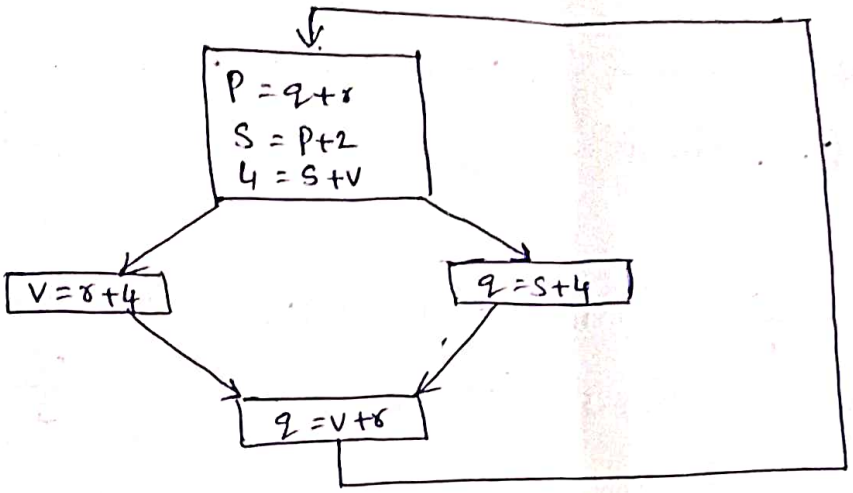
IN (B) → Set of variables live at begining of Block B

OUT (B) → Set of variable live just a ttes Block B

USE (B) → Variables that are used in Block B

DEF (B) → Variables that are assigned in block B

Ex: Consider the following flow graph and identify live variables in Block 2 and Block 3 collectively at Iteration 2



Node	USE [B]	DEF [B]	Iteration 1		Iteration 2	
			IN [B]	OUT [B]	IN [B]	OUT [B]
1	{q, r, v}	{P, S, 4}	{q, r, v}	{r, 4, S}	{r, q, 4}	{r, 4, S, v}
2	{r, 4}	{v}	{r, 4}	{v, r}	{r, 4}	{r, v}
3	{S, 4}	{q}	{S, 4}	{v, r}	{v, r, S, 4}	{r, v}
4	{v, r}	{q}	{v, r}	{q, r, 4}	{r, 4}	{r, q, v}

Iteration 1

$IN(B) = USE(B) \cup (OUT(B) - DEF(B))$

$IN(B_1) = USE(B_1) = \{q, r, v\}$

$IN(B_2) = USE(B_2) = \{r, 4\}$

$IN(B_3) = USE(B_3) = \{S, 4\}$

$IN(B_4) = USE(B_4) = \{v, r\}$

Iteration-1

$OUT[B_1]$ = Success $[B_1] \rightarrow B_2, B_3$ $\quad USE[B_2] \cup USE[B_3]$

$= \{r, 4, 8\}$ $\quad \{r, 4\} \cup \{s, 4\}$

$OUT[B_2]$ = Success $[B_4]$

$USE[B_4] = \{v, r\}$

$OUT[B_3]$ = Success $[B_4]$

$OUT[B_4] = \{v, r\}$

$OUT[B_4]$ = Success $[B_1]$

$OUT[B_1] = \{q, r, 4\}$

Iteration - 2

$IN[B_1] = USE[B_1] \cup (OUT[B_1] - DEF[B_1])$

$= \{q, r, 4\} \cup (\{r, 4, s\} - \{p, s, 4\})$

$IN[B_2] = USE[B_2] \cup (OUT[B_2] - DEF[B_2])$

$= \{r, 4\} \cup (\{s, v, r\} - \{v\})$

$= \{r, 4\} \cup \{r\} = \{r, 4\}$

$IN[B_3] = USE[B_3] \cup (OUT[B_3] - DEF[B_3])$

$= \{s, 4\} \cup (\{v, r\} - \{2\})$

$= \{s, 4\} \cup \{v, r\} = \{v, r, s, 4\}$

$IN[B_4] = USE[B_4] \cup (OUT[B_4] - DEF[B_4])$

$= \{v, r\} \cup (\{q, r, v\} - \{2\})$

$= \{v, r\} \cup \{r, v\}$

$= \{r, v\}$

Iteration - 2

$OUT(B_1)$ = Success B_2, B_3

$IN(B_2) \cup IN(B_3) = \{r, 4\} \cup \{v, r, s, 4\} = \{r, 4, v, s\}$

$OUT(B_2)$ = Success B_4

$IN(B_4) = \{r, v\}$

$OUT(B_3)$ = Success B_4

$IN(B_4) = \{r, v\}$

$OUT(B_4)$ = Success B_1

$IN(B_1) = \{r, q, v\}$

Conclusion :- The live variables in Block 2 and Block 3 Collectively at iteration is

formula $IN(B_2) \wedge IN(B_3)$

$= \{r, u\} \wedge \{r, s, v, u\} = \{r, u\}$

---

(3) Construct the target code for the following expression $x = (a-b) + (a-c) + (a-c)$ using simple code generator algorithm.

Sol:- **Step-1 :-** generate the address code

① $t = a-b$

② $u = a-c$

③ $v = t+u$

④ $d = v+u$

Note :- 'd' is live variable at the end

Step-2 :- Simple code generation

Statement	code generation	Register Descriptor	Address Descriptor.
		All registers are empty	
$t = a-b$	mov a, R0 SUB b, R0	R_0 contains t	t is in R_0
$u = a-c$	mov a, R_1 SUB b, R_1	R_0 contains t R_1 contains u	t is in R_0 u is in R_1
$v = t+u$	ADD R_1, R_2	R_0 contains v R_1 contains u	v is in R_0 u is in R_1
$d = v+u$	ADD R_1, R_0	R_0 contains d R_1 contains u	d is in R_0 u is in R_1

---

(4) Explain various address code representation in compiler with suitable examples.

Sol:- Quadruples : It contains 4 fields in its record structure

i. First field stores operator

ii. Second field stores operators or arguments

Lets say arg1, arg2

(iii). The last field

Ex :- $a = -b * d + c + (-b) * d$

1. $t_1 = -b$
2. $t_2 = t_1 * d$
3. $t_3 = c + t_2$
4. $t_4 = -b$
5. $t_5 = t_4 * d$
6. $t_6 = t_5 + t_3$
7. $a = t_6$

S.No	operator	Arg 1	Arg 2	Result
1	unary minus (-)	b		t_1
2	*	t_1	b	t_2
3	+	c	t_2	t_3
4	unary minus (-)	b		t_4
5	*	t_4	d	t_5
6	+	t_5	t_3	t_6
7	=	t_6		a

2. Triples : In representation it uses only 3 fields in its record structure

(i). The first fields represents operator

(ii). The next 2 fields represents arguments

Ex :- $a = -b * d + c + (-b) * d$

① $t_1 = -b$

② $t_2 = t_1 * b$

③ $t_3 = c + t_2$

④ $t_4 = -b$

⑤ $t_5 = t_4 * d$

⑥ $t_6 = t_3 + t_5$

⑦ $a = +b$

S.No	operator	Arg 1	Arg 2
1	Unary minus (-)	b	
2	*	(1)	d
3	+	c	(2)
4	Unary minus (-)	b	
5	*	(4)	d
6	+	(5)	(3)
7	=	a	(6)

3. Indirect_Triples : It uses pointess for listing the table.

Ex : $a = -b * d + c + (-b) * d$

1. $t_1 = -b$
2. $t_2 = t_1 * d$
3. $t_3 = c + t_2$
4. $t_4 = -b$
5. $t_5 = t_4 * d$
6. $t_6 = t_5 + t_3$
7. $a = t_6$

Points

Position	pointes	
1	10	→ 10
2	20	→ 20
3	30	→ 30
4	40	→ 40
5	50	→ 50
6	60	→ 60
7	70	→ 70

operator	Arg 1	Arg 2
Unary minus (-)	b	
*	10	d
+	c	20
Unary minus (-)	b	
*	40	d
+	50	30
=	a	60