Experiment No: \$10

	Aim: WAA to implement code generation.
	3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
	Theory woodA (short nop short
96	o aldocadable co
9 5	Roderngeneration is need to produce the target
	code for three address statements. It uses
	register to store the operands both 43 additional
0	statements. Insmpse +x91?
-	Consider 3 additional -statements to \$1 = 4+21
	It can have following I sequence of codes.
	Mov XIRO Sidato Lodare
	ADD Y. R. John prigouded
	object cede
•	Rogistos description contains the track of
	what is in currently each register.
1.50	Am address adescriptor lisanead toostore toot?
	location where current value of name can be
	inifound admiratione on a vom dos = +
	o91d 802
•	Code Generation Algorithm:
09	othe algorithmentakes a sequence of three 2 add resi
,9	statements 10 as input. For each three address
	statement of the form a: bopc perform
,9	withe various actions. so tooks ut to v
,5-	These actions gaves as of Follows
	1) Invoke of function getreg to find out the
09	or Plocation where the recult of From out ation o
Ro done	ni hop a should be stored vom

Experiment No : 318

.					1
Intermed	deveration	Opiec+	ot Af	W. MIE	•
code	generation	/ code	Ah	coluite con	le
		p 70 - s se co	-> RO	locadable	onda
solo o dla	001			locadable	
001 173	eed to produce	U 51 1	Ochoob	Sembles	CODE
	estmenstate.	229 rbbb	9964th	500 5 DOD	L L
2111600	Text con	3 NE 3 X	a45 a4	segiste.	
Chu	- Dot - I segment		2+r	atateme	()
206-	Head exapped of Text segment pata Lasegimen To	+ Iphorbibt	on &	1961200)	
_149000	Relocation Inf	Whollo!	3vod	11-14 +L	
# · · ·	Symbol Hable	2	0 1 X	100	
	Debugging in				
Jan	object node		_ 1	1 - 1 - 0	
1 2223	t-3 th aniatmos	o morta	4729 D	46 421 29 M	
tox dec	ly each segis	pagarus.		11- AND	-1
000 00	ode generated	Chedis 464	Derection	de to the Action	es descrip
- b .	mon de sulpte de	nesson	989/64	10000 to 1	D
	MovaiRe	Construction Co	SCHIMON USE VIE	n 4600 1	V 16-()
		! A	• 1	A - 1 - 5	
9- C	" Marcon Bross	Pal n	o dotad: 1	$N = \frac{3007}{100}$	· D
0.11.	# Mov son Appez	18.	n takes /4 =	DID EAIL	n Ko
	For 1991 & AKREE				
+ + 4	Dod : P ar	roi sar		Agmstemen	
	MOV LIRE	بر جورجاج ،			in Ri
	ARDORARDO				
VI du	boil ot partag	TO HOW	01 P	Y TOVOLY	in Ra
o por and	ADD RupRon	3 MTN 0 4	CONTRACTOR CO	ולפרמידוטה	0 0 4



Consult the address description Fory to detomin y1. If the value of y currently in memory and register both then prefor the register y'. If the value of y currently in memory and register both then prefer the register y'. If the value of y is not already in I there generates the information Mov y', L to place a copy of y in L. 3) Generate the infoduction OP 2', L where 2' is used to show the current location of z. If z is in both then prefer a register to a memory location. Update the address descriptor to n to indicate that n is in location L. IF n is in L then update its descriptor and remove a From all other descriptor. 4) of the current value of y have no next uses or not have an exit from black or in register then allows the register descriptor to indicate that offer execution of Mizyopz those register with no longer contains you no Conducion: In this, I learned and implemented

28 03 2024

Code:

```
op1, op2, op3, op4 = "", "", "",
print("Enter operation and operands (e.g., + o1 o2 o3): ")
while True:
 if not line:
 op1, op2, op3 = line[0], line[1], line[2]
   op4 = line[3]
   op4 = "Result"
 if op1 == "+":
  print(f"MOV AX, {op2}")
  print(f"MOV BX, {op3}")
  print("ADD AX, BX")
  print(f"MOV {op4}, AX")
 elif op1 == "-":
  print(f"MOV AX, {op2}")
  print(f"MOV BX, {op3}")
  print(f"MOV {op4}, AX")
  elif op1 == "*":
  print(f"MOV AX, {op2}")
  print(f"MOV BX, {op3}")
  print("MUL AX, BX")
  print(f"MOV {op4}, AX")
 elif op1 == "/":
  print(f"MOV AX, {op2}")
  print(f"MOV BX, {op3}")
  print(f"MOV {op4}, AX")
 elif op1 == "=":
   print(f"MOV {op2}, {op3}")
 else:
   print("Invalid operation")
print("\nCode generation successful")
```

Output:

```
Enter operation and operands (e.g., + o1 o2 o3):
- a b t
MOV AX, a
MOV BX, b
SUB AX, BX
MOV t, AX
– a c´u
MOV AX, a
MOV BX, c
SUB AX, BX
MOV u, AX
+ t u v
MOV AX, t
MOV BX, u
ADD AX, BX
MOV v, AX
+ v u d
MOV AX, v
MOV BX, u
ADD AX, BX
MOV d, AX
* v u k
MOV AX, v
MOV BX, u
MUL AX, BX
MOV k, AX
Code generation successful
```