COMPUTER ENGINEERING DEPARTMENT

SUBJECT: SYSTEM PROGRAMMING & COMPILER CONSTRUCTION

COURSE: T.E. **YEAR: 2020-2021** **SEMESTER: VI**
DEPT: COMPUTER ENGINEERING
SUBJECT CODE: CSC602 **EXAMINATION DATE: 04/06/2021**

==========

SYSTEM PROGRAMMING & COMPILER CONSTRUCTION
ANSWER SHEET

NAME : AMEY MAHENDRA THAKUR

SEAT NO. : 61021145

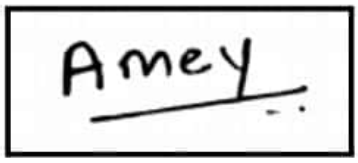
EXAM : SEMESTER VI

SUBJECT : SYSTEM PROGRAMMING & COMPILER CONSTRUCTION

DATE : 04-06-2021

DAY : FRIDAY

STUDENT SIGNATURE:



NAME: AMEY THAKUR
SUBJECT: SPCC
BRANCH: COMPUTER
EXAM: SEMESTER VI
SEAT NO.: 61021145
PAGE NO.: 1/9

Q2

A)

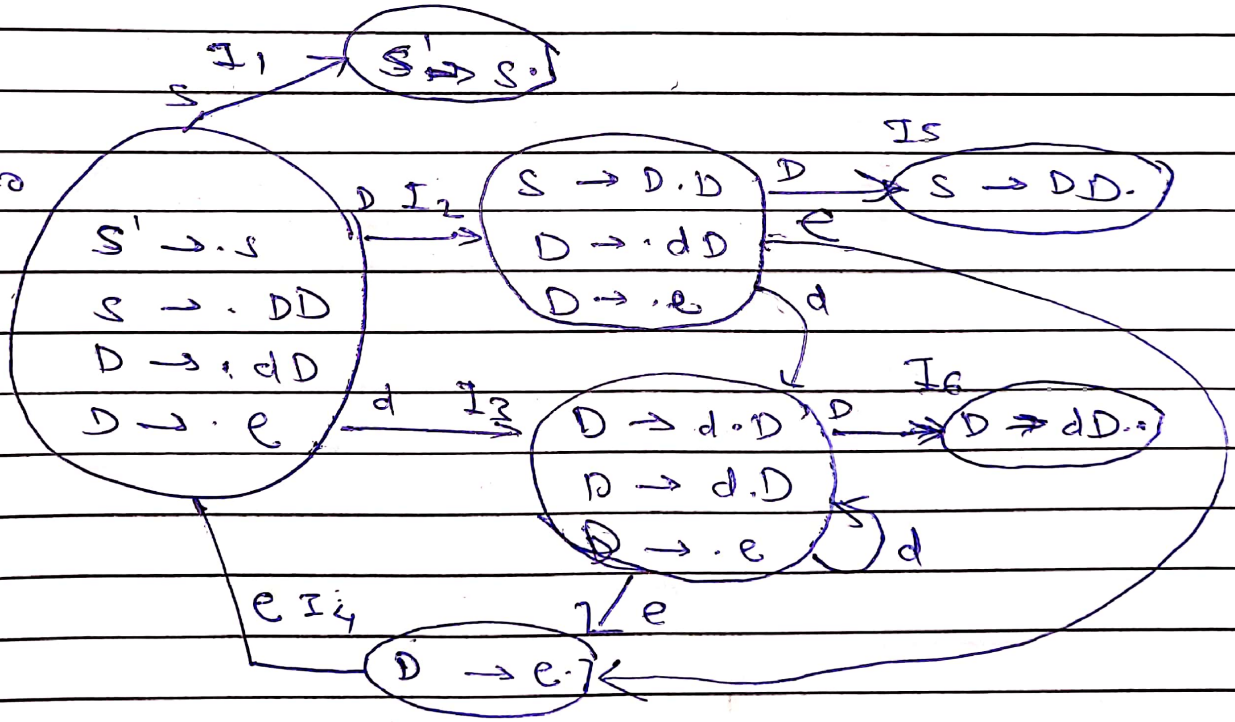
S → DD
D → dD
D → e

String : ddede

Step 1: Augment the grammar

S' → S
S → DD — (1)
D → dD — (2)
D → e — (3)

Step 2: Construct Goto Graph



04-06-2021

STUDENT SIGNATURE: Amey

Step 3: Design LR (0) Parser

	Action			Goto	
	d	e	$	S	D
0	S_3	S_4		1	2
1			Accept		
2	S_3	S_4			5
3	S_3	S_4			6
4	r_3	r_3	r_3		
5	r_1	r_1	r_1		
6	r_2	r_2	r_2		

Step 4: Algorithm

repeat forever
{
let 'x' be stack top number
let 'a' be input symbol
if m [x, a] = Accept then Accept and Break
else if m [x, a] = S_i then shift 'a' and push(i)
else if m [x, a] = r_j then reduce using r_j and perform GOTO
else Error ();
}

04-06-2021

STUDENT SIGNATURE: Amey

follow (S') = {$}
follow (S) = {$}
follow (D) = {e, $}

SLR Parsing Table.

	Actions			GOTO	
	d	e	$	S	D
0	S_3	S_4		1	2
1			Accept		
2	S_3	S_4			5
3	S_3	S_4			6
4		r_3	r_3		
5			r_1		
6			r_2		

String: ddede

Stack	Input	Action
$0	ddede$	shift d3
$0d3	dede$	shift d3
$0d3d3	ede$	shift e4
$0d3d3e4	de$	reduce D → e
$0d3D3D6	de$	Reduce D → dD
$0D2d3	de$	Reduce D → dD
$0D2d3e4	e$	shift e4
$0D2d3D6	$	Reduce D → e
$0D2D5	$	Reduce D → dD
$0S1	$	Reduce D → dD
S		Accept.

04-06-2021

STUDENT SIGNATURE: Amey

Q 2 B]

Single Pass assemblers:
- Single pass assembler is an assemblers which do it's all task in single scan.

Databases
(1) MOT
- MOT is organised and contains opcodes for assembly statements

(2) Symbol Table
- It is organized and includes all relevant information about the symbol defined and used in source program.

(3) Segment Register Table (SRTAB)
- Whenever assembler encounters ASSUME statement it stores the previous SRTAB on the stack and new SRTAB is created.

(4) Stored segment register table
- It stores Segment registers with segment names.

(5) Forward Reference ~~Problem~~ Table
- Information regarding forward referenced is put into FRT in a linked list manner

STUDENT SIGNATURE: Amey

04-06-2021

(5) CRT (Cross Reference Table)

- It is used for producing a cross reference director.
- This directory lists all reference to a symbol in the ascending order of statement number.

04-06-2021

STUDENT SIGNATURE: Amey

Q2.

C) Macro call.

MACRO

- In an assembly language program there can be group of instructions that may be repeated again and again.
- So we combine these instructions to a single group and assign a name to it which is called a MACRO.
- A MACRO is an extension to the basic ASSEMBLER language. They provide a means for generating a commonly used sequence of assembler instructions / statements.
- The sequence of instructions / statements will be coded ONE time within the macro definition. Whenever the sequence is needed within a program, the macro will be called.
- MACRO is defined as a single line abbreviation for a group of instruction.

Syntax:

Macro
Macro name

} macro body

MEND

04-06-2021 STUDENT SIGNATURE: Amey

- Execution is faster in macro. There is no transfer of control in Macro.

Features of Macro processor

1. Recognized the macro definition
2. Save macro definition
3. Recognize the macro call.
4. Perform macro expansion

04-06-2021 STUDENT SIGNATURE: Amey

Macro Expansion

Conditional Macro Expansion.

- The macro processor replaces each macro instruction with the corresponding group of source language statements. This is called macro expansion or expanding the macros.
- Conditional assembly are frequently considered to be mechanisms that allow a single version of the source code for a program to be used to generate multiple versions of the executable.
- Most macro processors can also modify the sequence of statements generated for a macro expansion, depending on the arguments supplied in the macro invocation. Conditional assembly is commonly used to describe this feature. It is also referred to as conditional macro expansion.
- Conditional assembly can be achieved with the help of AIF and AGO statements

(1) AIF Statement

- It is used to specify the branching condition
- This statement provides Conditional branching facility.
- Syntax : AIF (condition) Label
- If condition is satisfied the label is executed else it continue with the next execution
- It performs an arithmetic test and branches only if tested condition is true

STUDENT SIGNATURE: Amey

04-06-2021

② AGO Statement

- This statement provides the unconditional branching facility.
- We do not specify the condition
- Syntax : AGO. Label
- It specifies the label appearing on some other statement in the macro instruction definition
- The macro processor continues the sequential processing of instructions with the indicated statement.
- These statements are directives to the macro processor and do not appear in macro expansion.

How is macro different from subroutine

- Subroutines (FORM) can be called from both the program and are defined in other programs.
- Subroutines can take any amount of parameters whereas macros can take upto 9 parameters
- Subroutines are expanded at runtime whereas macros are expanded as compilation.

04-06-2024 STUDENT SIGNATURE: Amey