62020 SEM:06
Rebecca SPCC

BARFI Page No.
Date / /

1 /10

Q2]

c]

① A macro name is an abbreviation which stands for some related lines of code.
Macros are useful for the following purposes:
i) To simply and reduce the amount of repetitive coding
ii) To reduce error caused by repetitive coding
iii) To make an assembly program more readable

② A macro call leads to macro expansion. During macro expansion, the macro statement is replaced by sequence of assembly statements. A macro consists of a name, a set of formal parameters and a body of code.

③ A macro is a unit of specification for program generation through expansion

④ Macro Definition

eg:-

```
MACRO                                    start definition
INCR  &MEM_VAL, &INC_VAL, &REG           Name of macro
MOVER  &REG    &MEM_VAL
ADD    &REG    &INC_VAL                  } sequence of instructions
MOVEM  &REG    &MEM_VAL
MEND                                     End
```

INCR A,B ← macro call

Macro call : consists of a name optionally followed by an actual parameter list must be the same as the number of parameter. INCR A, B is an example of macro call used in the example here. If macro has no formal parameter list its call must have no actual parameter list.

Scanned with CamScanner

Q2]

⑤ Macro Expansion

- Replacement of macro call by corresponding sequence of instructions is called as macro expansion
- To expand a macro, the name of the macro is placed in the operation field and no special directives are necessary
- The assembly code sequence INCR A, B, AREG is an example of the macro call.
- The assembler recognizes INCR as the name of the macro expands the macro, and places a copy of the macro definition (along with the parameter substitutions)
- '+' sign in the preceding label field denote the expanded code.

Macro call

```
START 100
A   DS
B   DS
INCR  A,B,AREG
PRINT A
STOP
END
```

Macro definition

```
MACRO
INCR &A,&B,AREG
MOVER REG &A
ADD REG &B
MOVEM REG &A
MEND
```

Expanded code

```
START 100
A  DS  1
B  DS  1
MOVER REG A
ADD REG B
MOVEM REG A
PRINT A
STOP
END
```

A and B are actual parameters

&A and &B are formal parameters.

Scanned with CamScanner

BARFI | Page No. | Date

3 / 10

Q2]

(6) Macro name in the mnemonic field leads to expansion only in a subroutine name in call statement in the program leads to execution

(7) Macros are completely handled by the assembler during assembly time where as subroutines are completely handled by the hardware at runtime

(8) The hardware knows nothing about macros while the assembler knows nothing about subroutines

(9) The subroutine call instruction is assembled in the usual way and treated by the assembler as any other instruction

(10) Macro processing increases the size of the resulting code but results in faster expansion of program for expanded programs whereas for subroutines do not result into bulk object codes but has substantial overheads of control transfer during execution.

(11) Hardware executes the subroutine call instruction so, it has to know how to save the return address and how to branch to the subroutine.

Scanned with CamScanner

62020
SEM: 06 SPCC
Rebecca

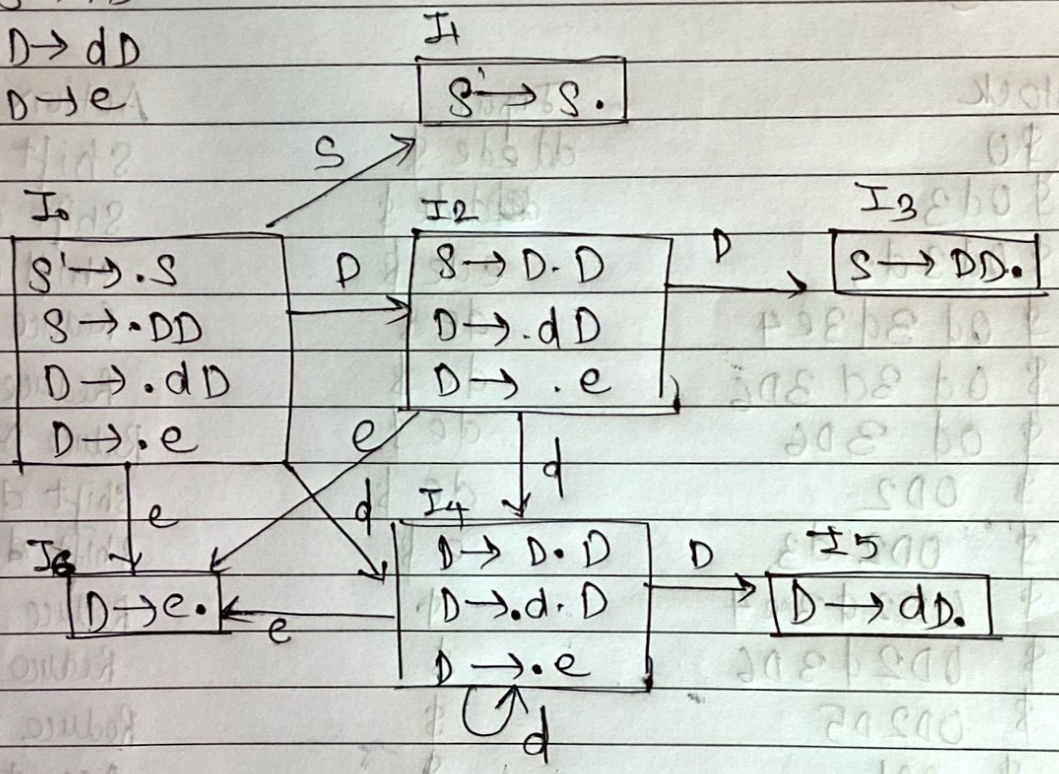
4/10

Q2]

A] String : ddede

$S' \to S$
$S \to DD$
$D \to dD$
$D \to e$

Step 1: Augument the grammar.



$S' = \{\$\}$
$S = \{\$\}$
$D = \{d, e, \$\}$

Design LR(0) parsing table

	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		
6	r_2	r_2	r_2		

Scanned with CamScanner

G2020
SEM: 06
SPCC

BARFI | Page No.
Date

5 / 10

~~Step 4. Algorithm.~~

SLR parsing table

String : ddede

Stack	Input	Action
$0	dd ede $	Shift d3
$0d3	ddede $	Shift d3
$0d3d3	ede $	Shift e4
$0d3d3e4	de $	Reduce D→e
$0d3d3D6	de $	Reduce D→dD
$0d3D6	de $	Reduce D→dD
$0D2	de $	Shift d3
$0D2d3	e $	Shift e4
$0D2d3e4	$	Reduce D→e
$0D2d3D6	$	Reduce D→dD
$0D2D5	$	Reduce D→dD
$0S1	$	Accept.

Algo

repeat forever
{
let 'x' be stack top number
let 'a' be input symbol
if M[x,a] = Accept then Accept & Break
else if M[x,a] = Si then Shift 'a' and Push 'i'
else if M[x,a] = rj then reduce using rj &
perform GOTO
else ERROR();
}

Scanned with CamScanner