INPUT TABLE -1 (OPCODE/MNEMONIC CODE )

|  |  |  |
| --- | --- | --- |
| Mnemonic opcode | Type of statement | (CODE,SIZE) |
| ADD | IS | (01,1) |
| SUB | IS | (02,1) |
| MULT | IS | (03,1) |
| MOVER | IS | (04,1) |
| MOVEM | IS | (05,1) |
| COMP | IS | (06,1) |
| BC | IS | (07,1) |
| DIV | IS | (08,1) |
| READ | IS | (09,1) |
| PRINT | IS | (10,1) |
| DS | DL | R#7(CODE:02) |
| DC | DL | R#8(CODE:01) |
| START | AD | 01 |
| END | AD | 02 |
| ORIGIN | AD | 03 |
| EQU | AD | 04 |
| LTORG | AD | 05 |

INPUT TABLE -2

|  |  |
| --- | --- |
| LE | 01 |
| LT | 02 |
| EQ | 03 |
| GT | 04 |
| GE | 05 |
| ANY | 06 |

(This 2 tables are pregenerated)

NOW,

We have to declare and initialize all the pointers like ,

Loc-cntr = 0 (default value)

Pooltab\_ptr = 1

Pooltab[1] = 1;

Littab\_ptr = 1 ;

1. START 200

Check c section in algorithm

Loc\_cntr = valuespecified in the operand field

Loc\_cntr = 200;

1. MOVER AREG,=’5’

MOVER is an imperative statement so go into the f section

Fetch the code and size of the MOVER and do entries in the table accordingly.

Code = 04

Loc\_cntr = loc\_cntr +1 = 200+1 =201

Now , check the operand is literal or symbol,

Here operand is literal . So we have to go into the if statement .

this\_literal = 5 ;

LITTAB[littab\_ptr] = 5 ;

|  |  |
| --- | --- |
| =’5’ |  |

LITTAB

Liitab\_ptr = littab\_ptr + 1 = 1+1 =2 ;

Generate IC = (IS,04)(1)(L,1)

1. MOVEM AREG , A

Movem is imperative statement so go into the f part in algorithm

Fetch, the code of MOVEM

Code = 05

Loc\_cntr = loc\_cntr + 1 = 201+1 =202

Check operand is literal or symbol

Here operand is symbol. So go into the else part and do entries accordingly .

|  |  |
| --- | --- |
| =’5’ |  |

|  |  |  |
| --- | --- | --- |
| A |  | 1 |

LITTAB SYMTAB

Generate IC = (IS,05)(1)(S,1)

1. LOOP MOVER AREG, A

In this statement found the lable first so , we have to do the entry of that lable in the symbol table.

And after that it is an imperative statement so we go into the f part and do entries in the appropriate table .

Code = 04(MOVER )

Loc\_cntr =loc\_cntr + 1 =202+1 =203

Check it operand is literal or symbol

Here is symbol , bt we have did the entry of symbol A before so we don’t do the entry again in the symbol table , only fetch the code of the symbol from the table

|  |  |
| --- | --- |
| =’5’ |  |
| =’5’ |  |

|  |  |  |
| --- | --- | --- |
| A |  | 1 |
| LOOP | 202 | 2 |

LITTAB SYMTAB

Generate IC = (IS,04)(1)(S,2)

1. MOVER CREG , B

It is imperative statement so go into the f part and do the appropriate entry.

Code = 04(MOVER)

Loc\_cntr = loc\_cntr + 1 =203+1 =204

We found the new symbol so do the entry in the symbol table

|  |  |
| --- | --- |
| =’5’ |  |

|  |  |  |
| --- | --- | --- |
| A |  | 1 |
| LOOP | 202 | 2 |
| B |  | 3 |

LITTAB

SYMTAB

Generate IC : (IS,04)(3)(S,3)

1. ADD CREG,=’1’

It is imperative statement so go into the f sec. of algorithm

Fetch the code of ADD

Code = 01(ADD)

Loc\_cntr = loc\_cntr + 1 = 203+1 = 204

Check the operand is literal or symbol

Here is literal so we have to do the entry in the literal table

this\_literal = 1 ;

LITTAB[littab\_ptr] = 1;

|  |  |
| --- | --- |
| =’5’ |  |
| =’1’ |  |

|  |  |  |
| --- | --- | --- |
| A |  | 1 |
| LOOP | 202 | 2 |
| B |  | 3 |

LITTAB SYMTAB

Liitab\_ptr = littab\_ptr + 1 = 2+1 =3 ;

Generate IC = (IS,01)(3)(L,2)

1. BC ANY,NEXT

It is an imperative statement so go into the f part of the algorithm

Fetch the code of the BC

Code = 07(BC)

Loc\_cntr= loc\_cntr + 1 = 204+1 =205

Now, we found the lable Next so go into the B part and do the entry of that label

|  |  |
| --- | --- |
| =’5’ |  |
| =’1’ |  |

|  |  |  |
| --- | --- | --- |
| A |  | 1 |
| LOOP | 202 | 2 |
| B |  | 3 |
| NEXT | 205 | 4 |

LITTAB SYMTAB

Generate IC = (IS,07)(6)(S,4)

1. LTORG

=’5’

=’1’

It is an Assemble directive so go to the B part of the algorithm

Allocate the address of the literals

=’5’ : 206

=’1’ : 207

|  |  |
| --- | --- |
| =’5’ | 206 |
| =’1’ | 207 |

LITTAB

|  |  |  |
| --- | --- | --- |
| A |  | 1 |
| LOOP | 202 | 2 |
| B |  | 3 |
| NEXT | 205 | 4 |

|  |
| --- |
| #1 |

POOLTAB

SYMTAB

Pooltab\_ptr = pooltab\_ptr+1 = 1+1 =2

Pooltab[pooltab\_ptr] = pooltab[2] = littab\_ptr = 3

So updated pooltab is

|  |
| --- |
| #1 |
| #3 |

POOLTAB

1. NEXT SUB AREG,=’1’

Here we found the lable first so check that label is present in the symbol table ,

If is it fetch the code of that label otherwise do the entry in the symbol table.

Next is present in the symbol table so fetch the entry code from there.

Code = 02(SUB)

Loc\_cntr = loc\_cntr + 1 = 207+1 =208

Check the operand is symbol or literal here is literal so we have to do the entry in the literal table .

|  |  |
| --- | --- |
| =’5’ | 206 |
| =’1’ | 207 |
| =’1’ |  |

|  |  |  |
| --- | --- | --- |
| A |  | 1 |
| LOOP | 202 | 2 |
| B |  | 3 |
| NEXT | 205 | 4 |

|  |
| --- |
| #1 |
| #3 |

LITTAB SYMTAB POOLTAB

Generae IC : (IS,02)(1)(L,3)

1. BC LT,BACK

It is an imperative statement so go into the f part of the algorithm

Fetch the code of the BC

Code = 07(BC)

Loc\_cntr= loc\_cntr + 1 = 208+1 =209

Now, we found the lable Next so go into the B part and do the entry of that label

|  |  |
| --- | --- |
| =’5’ | 206 |
| =’1’ | 207 |
| =’1’ |  |

|  |  |  |
| --- | --- | --- |
| A |  | 1 |
| LOOP | 202 | 2 |
| B |  | 3 |
| NEXT | 205 | 4 |
| Back | 209 | 5 |

|  |
| --- |
| #1 |
| #3 |

LITTAB POOLTAB

SYMTAB

Generate IC = (IS,07)(2)(S,5)

1. LAST STOP

Now, we found the lable Next so go into the B part and do the entry of that label

|  |  |
| --- | --- |
| =’5’ | 206 |
| =’1’ | 207 |
| =’1’ |  |

|  |  |  |
| --- | --- | --- |
| A |  | 1 |
| LOOP | 202 | 2 |
| B |  | 3 |
| NEXT | 205 | 4 |
| Back | 209 | 5 |
| LAST | 210 | 6 |

|  |
| --- |
| #1 |
| #3 |

LITTAB

SYMTAB POOLTAB

It is an imperative statement so we have to fetch the code of the STOP and increment the loc\_pntr

code = 00 (STOP)

Loc\_cntr = loc\_cntr + 1 =209+1 =210

Generate IC = (IS,00)(0)(S,6)

1. ORIGIN LOOP+2

We found the assemble directive so go into the C sec. of the algorithm

Loc\_cntr = value specified in the field

Here ORIGIN = LOOP + 2;

So, go into the symbol table and fetch the address of the loop table and add 2 in that address .

So ORIGIN = 203 + 2 = 205

Loc\_cntr = 205

1. MULT CREG , B

It is imperative statement so go into the f part and do the appropriate entry.

Code = 03(MULT)

Loc\_cntr = 205

B is already in the symbol table so we fetch the entry code of B from the symbol table

Generate IC : (IS,03)(3)(S,3)

Don’t do the any new entry in any table .

1. ORIGIN LAST+1

We found the assemble directive so go into the C sec. of the algorithm

Loc\_cntr = value specified in the field

Here ORIGIN = LAST + 1;

So, go into the symbol table and fetch the address of the loop table and add 2 in that address .

So ORIGIN = 210 + 1 = 211

Loc\_cntr = 211

1. A DS 1

It is a declaration statement. So go into the e part of the algorithm

Code = 02(DS)

Size = 1

Loc\_cntr = loc\_cntr +1 =211+1 = 212

Generate IC : (DL,2)(0)(S,1)

1. BACK EQU LOOP

We found the EQU directive in the statement

So go into the D part in the algorithm

Here the address field of the back is 209

But we found here BACK EQU LOOP so assign the address of back is same as loop.

|  |  |
| --- | --- |
| =’5’ | 206 |
| =’1’ | 207 |
| =’1’ |  |

|  |  |  |
| --- | --- | --- |
| A | 212 | 1 |
| LOOP | 203 | 2 |
| B | 213 | 3 |
| NEXT | 205 | 4 |
| Back | 202 | 5 |
| LAST | 210 | 6 |

|  |
| --- |
| #1 |
| #3 |

LITTAB

SYMTAB POOLTAB

1. B DS 1

It is a declaration statement. So go into the e part of the algorithm

Code = 02(DS)

Size = 1

Loc\_cntr = loc\_cntr +1 =212+1 = 213

Generate IC : (DL,2)(0)(S,3)

1. END

Process the B in the algorithm.

(AD,02)(0)(L,03)

|  |  |
| --- | --- |
| =’5’ | 206 |
| =’1’ | 207 |
| =’1’ | 214 |

LITTAB