

# Systems Software

## Chapter 1 Assembler

B.Tech. III (CO) Semester – 6	L	T	P	C
CO304 : SYSTEM SOFTWARE (CS-2)	3	1	2	5

- **ASSEMBLER**

Introduction, Cross Assembler, Micro Assembler, Meta Assembler, Single pass Assembler, Two Pass Assembler, Design of Operation code table, Symbol table, Literal table.

- **MACRO PROCESSOR**

Introduction of Macros, Macro processor design, Forward reference, Backward reference, positional parameters, keyword parameters, conditional assembly, Macro calls within Macros, Implementation of macros within Assembler.

Designing Macro name table, Macro Definition table, Keyword parameter table, Actual parameter table, Expansion time variable storage.

- **RUN TIME ENVIRONMENT**

- Absolute loader, Relocation - Relocating loader, Dynamic loader, Bootstrap loader, Linking-loader, Program relocatibility, Design of Absolute Loader, Design of direct-linking editor, other Loader scheme e.g. (Binders, Linking Loaders, Overlays, Dynamic Binders).

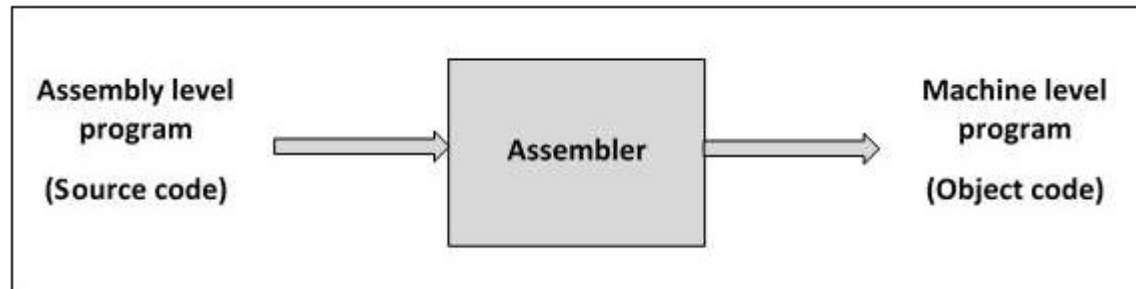
- **Books**

- 1) [D. M. Dhamdhere: “ System Programming and Operating System”.TMH](#)
- 2). A.V.Aho, R.Sethi & J D.Ullman, “Compilers-Principles, Techniques and Tools”. Pearson.
- 3). Leland L. Beck ,” System Software -An Introduction to System Programming”, 3/E, Addison Wesley,reprint 2003
- 4). Louden , Kenneth C :” Compiler Construction-Principles and Practice”,1/E, Thomson, 1997
- 5). Houlb : Compiler Design in C, PHI, EEE, 1995

# Assembler Definition

- An assembly language is a low-level programming language for microprocessors and other programmable devices.
- Each assembly language is specific to a particular computer architecture.
- Assembly language is machine dependent, low-level programming language which is specified to a certain computer system(or a family of computer system).

- Assembly language is converted into executable machine code by a **utility program (according to computer infrastructure)** referred to as an *assembler*.



# Types of Assembler

- 1. Cross Assembler**
- 2. Micro Assembler**
- 3. Meta Assembler**

# Cross Assembler

- A cross assembler is a program which generates machine code for a processor other than the one it is currently run on.
- An assembler is a program that converts assembly language into the actual binary processor specific machine code.
- Normally the machine code generated is for the processor used in the machine it is run on. A cross assembler takes this conversion process a step further by allowing you to generate machine code for a *different* processor than the one the compiler is run on.



- Cross-assembling facilitates the development of programs for systems that do not have the resources to support software development, such as an embedded system. In such a case, the resulting object code must be transferred to the target system, either via read-only memory (ROM, EPROM, etc.)

# Microassembler

- A microassembler is a computer program that helps prepare a microprogram, called *firmware*(permanent software programmed into a read-only memory.), to control the low level operation of a computer in much the same way an assembler helps prepare higher level code for a processor.
- The difference is that the microprogram is usually only developed by the processor manufacturer and works intimately with the computer hardware.
- The use of a microprogram allows the manufacturer to fix certain mistakes, including working around hardware design errors, without modifying the hardware.

# Meta Assembler

- **meta-assembler** A program that accepts the syntactic and semantic description of an assembly language, and generates an assembler for that language.

- **Basic three features for simplify programming**
- **Mnemonic operation codes(mnemonic opcodes ):**
  - Assembly language uses a mnemonic to represent each low-level machine instruction or operation
  - Eliminates to memorize numerical opcodes.
  - Provides helpful diagnostics.

- **Symbolic operands**

- Symbolic names can be associated with data or instructions
- It can also be used as operands in assembly language.
- Specify the data required by the operation
- Executable instructions can have zero to three operands
- Operands can be registers, memory variables, or constants

- **Data Declaration**

- Data can be declared in variety of notations
- Avoids manual conversion of constants into their internal machine representation.

# Statement Format

- Assembly language statement format
- [label] <Opcode> <Operand spec>[, <Operand spec>..]
- Anything Between this [..] is optional field.
- When label is there then it contains symbolic name

- <Operand spec> has following syntax:
- **<symbolic name>  
[+<displacement>][(index register)]**
- examples



- Statement Format:

- [Label] <Opcode> <operand spec> [,<operand spec>...]

- <operand spec> format:

- <Symbolic name> [+<displacement>][(<index register>)]

Ex. AREA+5(4)

Memory word with which  
name AREA is associated

Displacement /offset  
from AREA

Operand address is obtained by  
adding the contents of index  
register 4 to the address of AREA

- Example's specification is a combination of previous two.

Instruction opcode	Assembly mnemonic		
00	STOP		Stop execution
01	ADD	}	First operand is modified
02	SUB		
03	MULT		
04	MOVER		Register $\leftarrow$ memory move
05	MOVEM		Memory $\leftarrow$ register move
06	COMP		
07	BC		Branch on condition
08	DIV		Analog to SUB
09	READ	}	First operand is not used
10	PRINT		

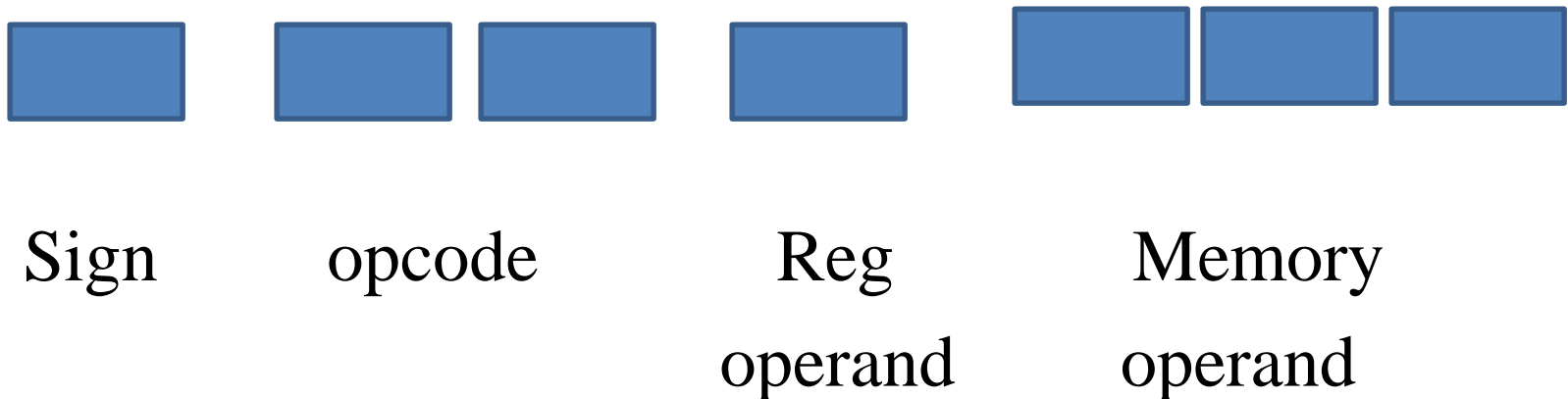
# Simple assembly languages

- In Assembly language each statement has 2 operands
  - First operand is always a register
  - Second is memory word using a symbolic name & optional displacement
  - *MOVER*: Second operand is source & first is target
  - *MOVEM*: opposite to *MOVER*
  - All arithmetic is performed in a register.
  - Condition code can be tested by a Branch on Condition (BC) instruction.
    - *BC* <condition code spec>, <memory address>
    - Transfers control to the memory word with memory address> if current value of condition code matches <condition code spec>.

- Conditional Code
- It can be tested by Branch on condition (BC) instruction.
- **Format:**
- **BC <conditional code spec> ,<memory address>**
- Conditional codes are LT,LE,EQ,GT,GE and ANY

# Machine instruction format

- Machine instruction format
- In this format all the addresses and constants are shown in decimal rather than octal or hexadecimal.



	START	101				
	READ	N	101)	+	09	0 113
	MOVER	BREG, ONE	102)	+	04	2 115
	MOVEM	BREG, TERM	103)	+	05	2 116
AGAIN	MULT	BREG, TERM	104)	+	03	2 116
	MOVER	CREG, TERM	105)	+	04	3 116
	ADD	CREG, ONE	106)	+	01	3 115
	MOVEM	CREG, TERM	107)	+	05	3 116
	COMP	CREG, N	108)	+	06	3 113
	BC	LE, AGAIN	109)	+	07	2 104
	MOVEM	BREG, RESULT	110)	+	05	2 114
	PRINT	RESULT	111)	+	10	0 114
	STOP		112)	+	00	0 000
N	DS	1	113)			
RESULT	DS	1	114)			
ONE	DC	'1'	115)			
TERM	DS	1	116)	+	00	0 001
	END					

- Assembly and equivalent machine language

# Assembly language statements

- Assembly program contains three kinds of statements
- Imperative statements
- Declaration Statements
- Assembler Directives

- **Imperative statements:**
- Actions to be performed during the execution of assembled program.
- Imperative statements represent machine instructions in symbolic form.
- For example, add means an addition instruction and mov denotes a data transfer instruction.



- **Declaration Statements:**
- Syntax of declaration statement

**[Label] DS < *constant* >**

**[Label] DC ' < *value* > '**

- DS (declare storage) :
  - Reserves area of memory and associated name with them

- Statements are

A	DS	1
G	DS	200

- First statement reserves 1 word with the name A.
- Second statement ?

- Statements are

A	DS	1
G	DS	200

- First statement reserves 1 word and associates the name A with it.
- Second statement reserves a block of 200 memory words. The name G is associated with the first word of the block.

- DC (Declare Constant)
  - It constructs memory words containing constants

XYZ                  DC                  ‘1’

- It assigns memory word containing the value ‘1’ to XYZ

- Constants
  - DC doesn't implement constants its only initialize block with the value.
  - This value can be changed
  - MOVEM CREG , XYZ
- Constants can be implemented in two ways
  - Immediate operands
  - Literals

- For ex

ADD AREG , 5

Note: simple assembly language doesn't support this ,but intel 8086 supports it.

- Literal
  - It is operand with =‘<value>’ syntax
  - Its differ from constant because its location cannot be specified in assembly program.
  - Its value cannot be changed during the execution of program

ADD	AREG , =‘5’	⇒	ADD AREG, FIVE
			-- --
			FIVE DC ‘5’

- Assembler Directives

- It instruct the assembler to perform certain action during program

START            <constants>

- Starts the target program with particular address

END

end of source program

# Advantages of assembly language:

- Use of symbolic operand specifications:
  - insertion of one statement in assembly program leads to changes in address of constants and reserved memory areas.
  - So addresses used in most instructions of the program had to change.
  - Such changes are not needed in assembly program.



	START	101					
	READ	N	101)	+	09	0	114
	MOVER	BREG, ONE	102)	+	04	2	116
	MOVEM	BREG, TERM	103)	+	05	2	117
AGAIN	MULT	BREG, TERM	104)	+	03	2	117
	MOVER	CREG, TERM	105)	+	04	3	117
	ADD	CREG, ONE	106)	+	01	3	116
	MOVEM	CREG, TERM	107)	+	05	3	117
	COMP	CREG, N	108)	+	06	3	114
	BC	LE, AGAIN	109)	+	07	2	104
	DIV	BREG, TWO	110)	+	08	2	118
	MOVEM	BREG, RESULT	111)	+	05	2	115
	PRINT	RESULT	112)	+	10	0	115
	STOP		113)	+	00	0	000
N	DS	1	114)				
RESULT	DS	1	115)				
ONE	DC	'1'	116)	+	00	0	001
TERM	DS	1	117)				
TWO	DC	'2'	118)	+	00	0	001
	END						

- Design Specifications of an assembler:
  - Identify the information necessary to perform a task
  - Design a suitable data structure to record the information
  - Determine the processing necessary to obtain and maintain the information
  - Determine the processing necessary to perform a task
- *The fundamental info requirements arise in the synthesis phase.*

# Synthesis Phase

Consider the assembly statement

MOVER            BREG, ONE

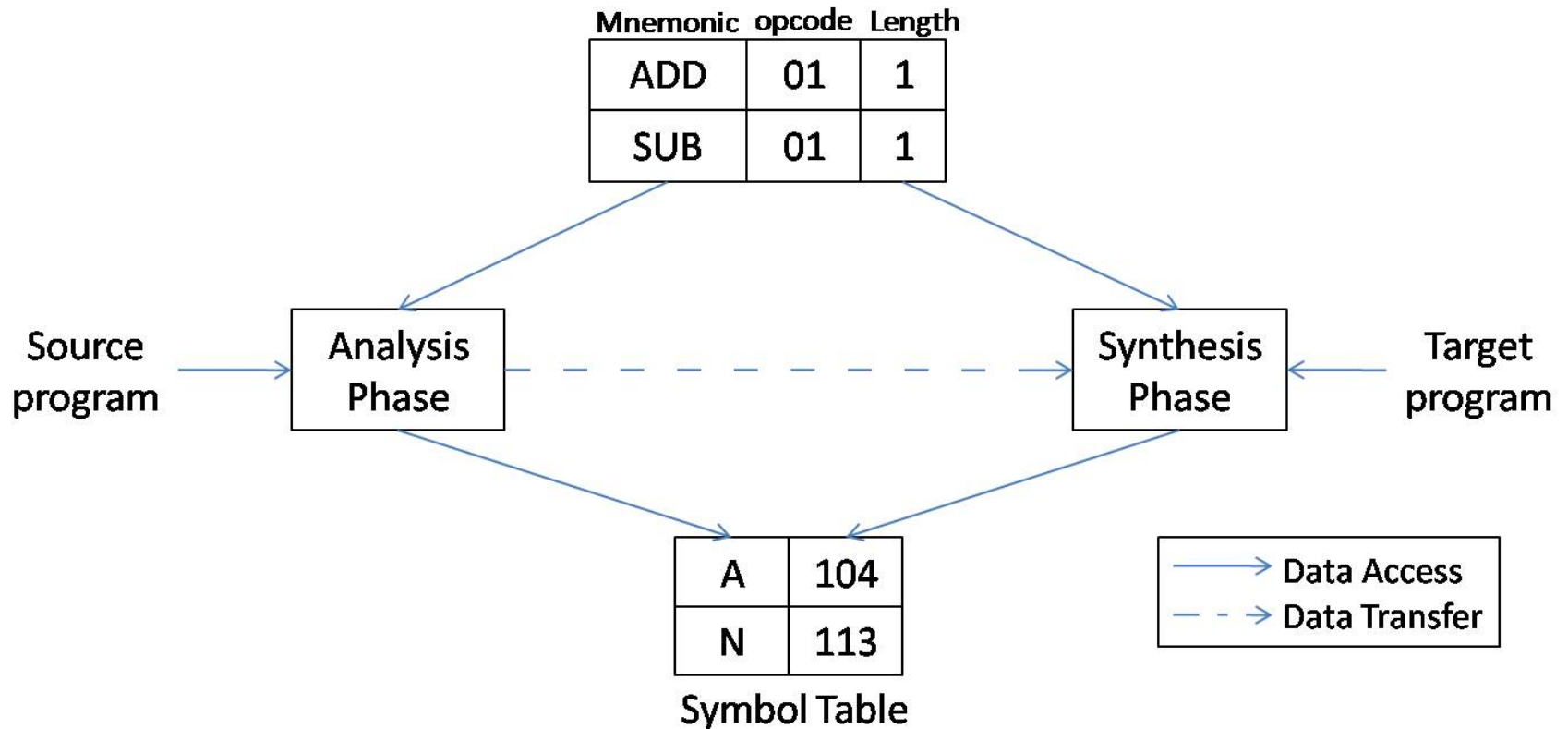
- We must have the following information to synthesize the machine instruction
  - Address of the memory word with which ONE is associated
    - » It depends on source program
    - » So, it must be made available by analysis phase.
  - Machine operation code corresponding to the mnemonic MOVER
    - Does not depend on source program, but depend upon assembly language
    - Hence the synthesis phase can determine this information itself.

- During synthesis phase, two data structures are used:-
  - Symbol Table
    - symbol table's each entry has two fields:-
      - **name and address.**
    - This table is built by analysis phase and used during synthesis.
  - Mnemonics Table
    - An entry in the mnemonics table has two fields:-
      - **mnemonic and opcode.**
    - fixed table and merely accessed by analysis and synthesis phases.
- By using symbol table, synthesis phase obtains the machine address with which name is associated.
- By using Mnemonics table, synthesis phase obtains machine opcode corresponding to a mnemonic.

# Analysis phase

- The primary function performed by the analysis phase is the building of the symbol table.
  - For this, it must determine the addresses with which symbolic names used in a program are associated.
  - Some addresses can be determined directly (Eg. Address of first instruction in program) while some other must be inferred. (Eg. In prog. for determining address of N, we must fix address of all program elements preceding it.) This is called **memory allocation**.

- **Location Counter**
  - Location Counter (LC) is a data structure which use to implement memory allocation
  - The location counter contains the address of the next memory word in the target program.
  - LC is initialized to the constant specified in the START statement.
- To update the contents of LC, analysis phase needs to know the lengths of different instructions.
  - This info. Depends on the assembly language.
  - To include this Mnemonic table can be extended and a new field called as length is used
- The processing involved to maintain the LC is called **LC processing**.



## Data Structure of an Assembler

- Tasks performed by Analysis Phases:-
  1. Isolate the label, mnemonic opcode and operand fields of a statement.
  2. If a label is present, enter the pair (symbol,<LC contents>) in a new entry of symbol table.
  3. Check validity of mnemonic opcode by look-up in Mnemonics table.
  4. Perform LC processing i.e. update value contained in LC.



- Tasks performed by Synthesis Phase:-
  1. Obtain the machine opcode corresponding to mnemonics from mnemonics table.
  2. Obtain address of memory operand from symbol table.
  3. Synthesize the machine form of a constant, if any.

# Pass Structure of Assemblers

- The pass of a language processor is one complete scan of the source program.
- There are mainly two assembly schemes:-
  1. Two pass assembly scheme
  2. One pass assembly scheme.

# Two Pass Translator / Two Pass Assembler

- can handle forward references easily.
- The LC processing is performed in the first pass and symbols defined in the program are entered into the symbol table in this pass.
- The first pass performs analysis of the source program while the second pass performs the synthesis of the target program.
- The first pass constructs an intermediate representation (IR) of the source program for use by the second pass.
  - This representation consists of two main components
    - data structures e.g. Symbol table
    - a processed form of source program. Also called as Intermediate Code (IC).

# Single Pass Translation / Single Pass Assembler

- In a single pass assembler, the LC processing and construction of symbol table proceeds as in two pass assembler.
- The problem of forward references is tackled using a process called back patching.
- Initially, operand field of an instruction containing a forward reference is left blank.
- When the definition of forward referenced symbol is encountered, its address is put into this field which is left blank initially.

- ONE is a forward reference.
- The instruction opcode and address of BREG will be assembled to reside in location 101 and the insertion of second operand's address at a later stage can be indicated by adding an entry of the Table of Incomplete Instructions (TII).
  - This entry is a pair (<instruction address>, <symbol>).
    - e.g. (101, ONE) in this case.
- By the time, the END statement is processed, the symbol table would contain the addresses of all symbols defined in the source program and TII would contain info. describing forward references.

# **Design of a Two Pass Assembler**

- **PASS 1:-**

1. Separate the symbol, mnemonic opcode and operand fields.
2. Build the symbol table.
3. Perform LC processing.
4. Construct intermediate representation

- **PASS 2:-**

- **1. synthesize the target program**

- The Pass 1 performs analysis of the source program and synthesis of the intermediate representation
- while Pass 2 processes the intermediate representation (IR) to synthesize the target program.

# Advanced Assembler Directives

- **(i) ORIGIN:-**

- The syntax of this directive is

**ORIGIN      <address spec>**

where <address spec> is an <operand spec> or <constant>.

- This directive indicates that LC should be set to the address given by <address spec>.
- The ‘ORIGIN’ statement is useful when the target program does not consist of consecutive memory words. The ability to use an <address spec> in the ORIGIN statement provides the ability to perform LC processing in a relative manner rather than absolute manner.



1		START	200			
2		MOVER	AREG, = '5'	200)	+04	1 211
3		MOVEM	AREG, A	201)	+05	1 217
4	LOOP	MOVER	AREG, A	202)	+04	1 217
5		MOVER	CREG, B	203)	+05	3 218
6		ADD	CREG, = '1'	204)	+01	3 212
7		...				
12		BC	ANY, NEXT	210)	+07	6 214
13		LTORG				
			= '5'	211)	+00	0 005
			= '1'	212)	+00	0 001
14		...				
15	NEXT	SUB	AREG, = '1'	214)	+02	1 219
16		BC	LT, BACK	215)	+07	1 202
17	LAST	STOP		216)	+00	0 000
18		ORIGIN	LOOP+2			
19		MULT	CREG, B	204)	+03	3 218
20		ORIGIN	LAST+1			
21	A	DS	1	217)		
22	BACK	EQU	LOOP			
23	B	DS	1	218)		
24		END				
25			= '1'	219)	+00	0 001

- **(ii) EQU:-**

- The syntax of this directive is

- [symbol] EQU [address spec]**

- where [address spec] is an [operand spec] or [constant].

- The EQU statement defines the symbol to represent [address spec].
  - BACK EQU LOOP

- **LTORG:-**

- *The LTORG statement permits a programmer to specify where literals should be placed.*
- *By default, assembler places the literals after the END statement.*
- *At every LTORG statement, the assembler allocates memory to the literals of a literal pool.*
  - *This pool contains all the literals used in the program.*

# Pass 1 of the Assembler

- Pass 1 uses the following data structures:
  - **OPTAB:** - A table of mnemonic opcodes and related info.
  - **SYMTAB:** - Symbol Table. (contains the fields address and length)
  - **LITTAB:** - A table of literals used in the program.(contains the fields literal and address.)

- **OPTAB: Static table:** The content will never change
  - contains the fields mnemonic opcode, class and mnemonic information.
  - The ‘class’ field indicates whether the opcode corresponds to
    - an imperative statement (IS),
    - a declaration statement (DL) or
    - an assembler directive (AD)
  - If an imperative statement is present,
    - then the mnemonic info. field contains the pair (machine opcode, instruction length)
  - else it contains the pair id of a routine to handle the declaration or directive statement.

- In pass 1, OPTAB is used to look up and validate mnemonics in the source program.
  - In pass 2, OPTAB is used to translate mnemonics to machine instructions.
- **SYMTAB**
  - Include the label name and value (address) for each label in the source program
  - Dynamic table (I.e., symbols may be inserted, deleted, or searched in the table)

- The LITTAB is used to collect all literals used in the program. The awareness of different literal pools are maintained by an auxiliary table POOLTAB. This table contains the literal no. of starting literal of each literal pool.

<i>mnemonic opcode</i>	<i>class</i>	<i>mnemonic info</i>
MOVER	IS	(04,1)
DS	DL	R#7
START	AD	R#11
	:	

OPTAB

<i>symbol</i>	<i>address</i>	<i>length</i>
LOOP	202	1
NEXT	214	1
LAST	216	1
A	217	1
BACK	202	1
B	218	1

SYMTAB

	<i>literal</i>	<i>address</i>
1	= 'S'	
2	= 'I'	
3	= 'I'	

LITTAB

<i>literal no</i>
#1
#3
—

POOLTAB

## Data Structure of Assembler pass 1



# Intermediate Code forms

- The intermediate code consists of a set of IC units, each IC unit consisting of following three fields: -
  1. Address
  2. Representation of mnemonic op code
  3. Representation of operands



IC unit

- There are generally two criteria for choice of intermediate code (IC)
  - processing efficiency
  - memory economy.
- Arise of some variants forms of intermediate code mainly operand and address fields is due to trade off between processing efficiency and memory economy.

- The mnemonic field contains a pair of the form (statement class, code)
- Here, statement class can be one of the imperative (IS) (DL) (AD).
  - For (IS), code is the instruction opcode in machine language.
  - For DL and AD, code is an ordinal number within class.

<u>Declaration Statements</u>		<u>Assembler Directives</u>	
DC	01	START	01
DS	02	END	02
		ORIGIN	03
		EQU	04
		LTORG	05

# **Intermediate Code for Imperative Statement**

## Variant I:

- The first operand is represented by a single digit number which is a code for register (i.e. 1-4 for AREG – DREG) or condition code itself (1-6 for LT-ANY)
- The second operand, which is memory operand, is represented by a pair of form (operand class, code) where operand class is one of C, S and L standing for constant, symbol and literal resp.
- For a constant, code field contains internal representation of constant itself.
- For symbol or literal, code field contains the ordinal no. of operand's entry in SYMTAB or LITTAB.

	START	200	(AD,01)	(C,200)
	READ	A	(IS,09)	(S,01)
LOOP	MOVER	AREG, A	(IS,04)	(1)(S,01)
	:		:	
	SUB	AREG, ='1'	(IS,02)	(1)(L,01)
	BC	GT, LOOP	(IS,07)	(4)(S,02)
	STOP		(IS,00)	
A	DS	1	(DL, 02)	(C,1)
	LTORG		(DL,05)	
	...		...	

## Intermediate Code- Variant I

## **Variant II:**

- This variant differs from variant 1 in that the operand fields of source statements are replaced by their processed forms.
- For declarative statements and assembler directives, processing of operand fields contain processed forms.
- For imperative statements, the operand field is processed only to identify literal references.

- Variant 2 reduces the work of Pass 1 by transferring burden of operand processing from Pass 1 to pass 2.

	START	200	(AD,01)	(C,200)
	READ	A	(IS,09)	A
LOOP	MOVER	AREG, A	(IS,04)	AREG, A
	:		:	
	SUB	AREG, ='1'	(IS,02)	AREG, (L,01)
	BC	GT, LOOP	(IS,07)	GT, LOOP
	STOP		(IS,00)	
A	DS	1	(DL,02)	(C,1)
	LTORG		(DL,05)	
	...		...	



	START	100	
	MOVER	AREG,A	
LOOP	PRINT	B	
	ADD	BREG, =‘9’	
	SUB	BREG, D	
	COMP	CREG, =‘23’	
	LTORG		
A	DS	3	
LABEL	EQU	LOOP	
	ORIGIN	500	
L1	MULT	CREG, =‘7’	
	SUB	BREG, =‘93’	
	LTORG		
B	DC	‘10’	
	MOVEM	CREG, =‘7’	
	PRINT	=‘7’	
D	DC	‘8’	
	END		

	START	100	
	MOVER	AREG,A	
LOOP	PRINT	B	
	ADD	BREG, ='9'	
	SUB	BREG, D	
	COMP	CREG, ='23'	
	LTORG		
A	DS	3	
LABEL	EQU	LOOP	
	ORIGIN	500	
L1	MULT	CREG, ='7'	
	SUB	BREG, ='93'	
	LTORG		
B	DC	'10'	
	MOVEM	CREG, ='7'	
	PRINT	= '7'	
D	DC	'8'	
	END		

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ

	START	100	
	MOVER	AREG,A	
LOOP	PRINT	B	
	ADD	BREG, ='9'	
	SUB	BREG, D	
	COMP	CREG, ='23'	
	LTORG		
A	DS	3	
LABEL	EQU	LOOP	
	ORIGIN	500	
L1	MULT	CREG, ='7'	
	SUB	BREG, ='93'	
	LTORG		
B	DC	'10'	
	MOVEM	CREG, ='7'	
	PRINT	= '7'	
D	DC	'8'	
	END		

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02

	START	100	
	MOVER	AREG,A	
LOOP	PRINT	B	
	ADD	BREG, ='9'	
	SUB	BREG, D	
	COMP	CREG, ='23'	
	LTORG		
A	DS	3	
LABEL	EQU	LOOP	
	ORIGIN	500	
L1	MULT	CREG, ='7'	
	SUB	BREG, ='93'	
	LTORG		
B	DC	'10'	
	MOVEM	CREG, ='7'	
	PRINT	= '7'	
D	DC	'8'	
	END		

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START	100	
MOVER	AREG,A	
LOOP	PRINT	B
ADD	BREG, =‘9’	
SUB	BREG, D	
COMP	CREG, =‘23’	
LTORG		
A	DS	3
LABEL	EQU	LOOP
ORIGIN	500	
L1	MULT	CREG, =‘7’
SUB	BREG, =‘93’	
LTORG		
B	DC	‘10’
MOVEM	CREG, =‘7’	
PRINT	=‘7’	
D	DC	‘8’
END		

SYM_NO	SYMBOL	ADDRESS

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

	START	100	
	MOVER	AREG,A	
LOOP	PRINT	B	
	ADD	BREG, ='9'	
	SUB	BREG, D	
	COMP	CREG, ='23'	
	LTORG		
A	DS	3	
LABEL	EQU	LOOP	
	ORIGIN	500	
L1	MULT	CREG, ='7'	
	SUB	BREG, ='93'	
	LTORG		
B	DC	'10'	
	MOVEM	CREG, ='7'	
	PRINT	= '7'	
D	DC	'8'	
	END		

SYM_NO	SYMBOL	ADDRESS

LIT_NO	LITERAL	ADDRESS

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START    100			(AD,01) – (C,100)
MOVER AREG,A			
LOOP	PRINT	B	
ADD    BREG, ='9'			
SUB    BREG, D			
COMP   CREG, ='23'			
LTORG			
A	DS	3	
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, ='7'	
SUB    BREG, ='93'			
LTORG			
B	DC	'10'	
MOVEM CREG, ='7'			
PRINT    ='7'			
D	DC	'8'	
END			

SYM_NO	SYMBOL	ADDRESS

LIT_NO	LITERAL	ADDRESS

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START    100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	
ADD    BREG, ='9'			
SUB    BREG, D			
COMP   CREG, ='23'			
LTORG			
A	DS	3	
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, ='7'	
SUB    BREG, ='93'			
LTORG			
B	DC	'10'	
MOVEM CREG, ='7'			
PRINT    ='7'			
D	DC	'8'	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	

LIT_NO	LITERAL	ADDRESS

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04



START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			
SUB BREG, D			
COMP CREG, =‘23’			
LTORG			
A	DS	3	
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	
SUB BREG, =‘93’			
LTORG			
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	
2	LOOP	101
3	B	

LIT_NO	LITERAL	ADDRESS

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			
COMP CREG, =‘23’			
LTORG			
A	DS	3	
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	
SUB BREG, =‘93’			
LTORG			
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	
2	LOOP	101
3	B	

LIT_NO	LITERAL	ADDRESS
1	=‘9’	

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, ='9'			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, ='23'			
LTORG			
A	DS	3	
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, ='7'	
SUB BREG, ='93'			
LTORG			
B	DC	'10'	
MOVEM CREG, ='7'			
PRINT ='7'			
D	DC	'8'	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	
2	LOOP	101
3	B	
4	D	

LIT_NO	LITERAL	ADDRESS
1	= '9'	

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, ='9'			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, ='23'			104) (IS,08) (3)(L,02)
LTORG			
A	DS	3	
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, ='7'	
SUB BREG, ='93'			
LTORG			
B	DC	'10'	
MOVEM CREG, ='7'			
PRINT ='7'			
D	DC	'8'	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	
2	LOOP	101
3	B	
4	D	

LIT_NO	LITERAL	ADDRESS
1	= '9'	
2	= '23'	

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
A	DS	3	
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	
SUB BREG, =‘93’			
LTORG			
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	
2	LOOP	101
3	B	
4	D	

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	
SUB BREG, =‘93’			
LTORG			
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	
2	LOOP	101
3	B	
4	D	

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	
SUB BREG, =‘93’			
LTORG			
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	
4	D	

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	
SUB BREG, =‘93’			
LTORG			
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	
4	D	
5	LABEL	101

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04



START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			
LTORG			
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	
4	D	
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			501) (IS,04) (2)(L,04)
LTORG			
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	
4	D	
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	
4	=‘93’	

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			501) (IS,04) (2)(L,04)
LTORG			502) (AD,05) – 007
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	
4	D	
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			501) (IS,04) (2)(L,04)
LTORG			502) (AD,05) – 007
			503) (AD,05) – 093
B	DC	‘10’	
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	
4	D	
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			501) (IS,04) (2)(L,04)
LTORG			502) (AD,05) – 007
			503) (AD,05) – 093
B	DC	‘10’	504) (DL,02) – 010
MOVEM CREG, =‘7’			
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			501) (IS,04) (2)(L,04)
LTORG			502) (AD,05) – 007
			503) (AD,05) – 093
B	DC	‘10’	504) (DL,02) – 010
MOVEM CREG, =‘7’			505) (IS,02) (3)(L,05)
PRINT =‘7’			
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			501) (IS,04) (2)(L,04)
LTORG			502) (AD,05) – 007
			503) (AD,05) – 093
B	DC	‘10’	504) (DL,02) – 010
MOVEM CREG, =‘7’			505) (IS,02) (3)(L,05)
PRINT =‘7’			506) (IS,09) – (L,05)
D	DC	‘8’	
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			501) (IS,04) (2)(L,04)
LTORG			502) (AD,05) – 007
			503) (AD,05) – 093
B	DC	‘10’	504) (DL,02) – 010
MOVEM CREG, =‘7’			505) (IS,02) (3)(L,05)
PRINT =‘7’			506) (IS,09) – (L,05)
D	DC	‘8’	507) (DL,02) – 008
END			

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04



START 100			(AD,01) – (C,100)
MOVER AREG,A			100) (IS,01) (1)(S,01)
LOOP	PRINT	B	101) (IS,09) – (S,03)
ADD BREG, =‘9’			102) (IS,03) (2)(L,01)
SUB BREG, D			103) (IS,04) (2)(S,04)
COMP CREG, =‘23’			104) (IS,08) (3)(L,02)
LTORG			105) (AD,05) – 009
			106) (AD,05) – 023
A	DS	3	107) (DL,01) – 03
LABEL	EQU	LOOP	
ORIGIN 500			
L1	MULT	CREG, =‘7’	500) (IS,05) (3)(L,03)
SUB BREG, =‘93’			501) (IS,04) (2)(L,04)
LTORG			502) (AD,05) – 007
			503) (AD,05) – 093
B	DC	‘10’	504) (DL,02) – 010
MOVEM CREG, =‘7’			505) (IS,02) (3)(L,05)
PRINT =‘7’			506) (IS,09) – (L,05)
D	DC	‘8’	507) (DL,02) – 008
END			508) (AD,02) – 000

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

(MOT)	
Op-Code	Mnemonics
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MULT
06	DIV
07	BC
08	COMP
09	PRINT
10	READ
DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	
100) (IS,01) (1)(S,01)	
101) (IS,09) – (S,03)	
102) (IS,03) (2)(L,01)	
103) (IS,04) (2)(S,04)	
104) (IS,08) (3)(L,02)	
105) (AD,05) – 009	
106) (AD,05) – 023	
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	
101) (IS,09) – (S,03)	
102) (IS,03) (2)(L,01)	
103) (IS,04) (2)(S,04)	
104) (IS,08) (3)(L,02)	
105) (AD,05) – 009	
106) (AD,05) – 023	
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	
102) (IS,03) (2)(L,01)	
103) (IS,04) (2)(S,04)	
104) (IS,08) (3)(L,02)	
105) (AD,05) – 009	
106) (AD,05) – 023	
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	
103) (IS,04) (2)(S,04)	
104) (IS,08) (3)(L,02)	
105) (AD,05) – 009	
106) (AD,05) – 023	
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	
104) (IS,08) (3)(L,02)	
105) (AD,05) – 009	
106) (AD,05) – 023	
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	
105) (AD,05) – 009	
106) (AD,05) – 023	
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	
106) (AD,05) – 023	
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509



INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	502) 00 0 007
503) (AD,05) – 093	
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	502) 00 0 007
503) (AD,05) – 093	503) 00 0 093
504) (DL,02) – 010	
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	502) 00 0 007
503) (AD,05) – 093	503) 00 0 093
504) (DL,02) – 010	504) 00 0 010
505) (IS,02) (3)(L,05)	
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509



INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	502) 00 0 007
503) (AD,05) – 093	503) 00 0 093
504) (DL,02) – 010	504) 00 0 010
505) (IS,02) (3)(L,05)	505) 02 3 509
506) (IS,09) – (L,05)	
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	502) 00 0 007
503) (AD,05) – 093	503) 00 0 093
504) (DL,02) – 010	504) 00 0 010
505) (IS,02) (3)(L,05)	505) 02 3 509
506) (IS,09) – (L,05)	506) 09 0 509
507) (DL,02) – 008	
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	502) 00 0 007
503) (AD,05) – 093	503) 00 0 093
504) (DL,02) – 010	504) 00 0 010
505) (IS,02) (3)(L,05)	505) 02 3 509
506) (IS,09) – (L,05)	506) 09 0 509
507) (DL,02) – 008	507) 00 0 008
508) (AD,02) – 000	

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	502) 00 0 007
503) (AD,05) – 093	503) 00 0 093
504) (DL,02) – 010	504) 00 0 010
505) (IS,02) (3)(L,05)	505) 02 3 509
506) (IS,09) – (L,05)	506) 09 0 509
507) (DL,02) – 008	507) 00 0 008
508) (AD,02) – 000	508) 00 0 000

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

INTERMEDIATE CODE	TARGET CODE
(AD,01) – (C,100)	01 0 100
100) (IS,01) (1)(S,01)	100) 01 1 107
101) (IS,09) – (S,03)	101) 09 0 504
102) (IS,03) (2)(L,01)	102) 03 2 105
103) (IS,04) (2)(S,04)	103) 04 2 507
104) (IS,08) (3)(L,02)	104) 08 3 106
105) (AD,05) – 009	105) 00 0 009
106) (AD,05) – 023	106) 00 0 023
107) (DL,01) – 03	107) 00 0 003
500) (IS,05) (3)(L,03)	500) 05 3 502
501) (IS,04) (2)(L,04)	501) 04 2 503
502) (AD,05) – 007	502) 00 0 007
503) (AD,05) – 093	503) 00 0 093
504) (DL,02) – 010	504) 00 0 010
505) (IS,02) (3)(L,05)	505) 02 3 509
506) (IS,09) – (L,05)	506) 09 0 509
507) (DL,02) – 008	507) 00 0 008
508) (AD,02) – 000	508) 00 0 000
	509) 00 0 007

SYM_NO	SYMBOL	ADDRESS
1	A	107
2	LOOP	101
3	B	504
4	D	507
5	LABEL	101
6	L1	500

LIT_NO	LITERAL	ADDRESS
1	=‘9’	105
2	=‘23’	106
3	=‘7’	502
4	=‘93’	503
5	=‘7’	509

SOURCE CODE	INTERMEDIATE CODE
START 101	
READ N	
MOVER BREG, ONE	
MOVEM BREG, TERM	
AGAIN MULT BREG, TERM	
MOVER CREG, TERM	
ADD CREG, ONE	
MOVEM CREG, TERM	
COMP CREG, N	
BC LE, AGAIN	
MOVEM BREG, RESULT	
PRINT RESULT	
STOP	
N DS 1	
RESULT DS 1	
ONE DC '1'	
TERM DS 1	
END	

DL	
DC	01
DS	02
AD	
START	01
END	02
ORIGIN	03
EQU	04
LTORG	05

(MOT)	
Op-Code	Mnemonics
00	STOP
01	ADD
02	SUB
03	MULT
04	MOVER
05	MOVEM
06	COMP
07	BC
08	DIV
09	READ
10	PRINT