A ONE-PASS ASSEMBLY PROGRAM

FOR THE LGP-30 COMPUTER

By

T. D. Lassagne

Captain USAF

SC0580

1. INTRODUCTION

If you want to get your money's worth from a computer system, one of your basic requirements is an assembly program that offers you speed, flexibility, and convenience. An assembler that truly "pays its way" will use a minimum of machine time and its language will be powerful enough to make programming fast, accurate, and free from drudgery. These were the criteria that guided the development of this assembly program.

First, it had to be fast. One good way to gain speed on the LGP-30 is to minimize the amount of input and output. For example, by suppressing the assembly listing we can speed the task up somewhat. But by performing the assembly in one pass instead of two, we get a substantial increase in speed. The source tape has to be read only once, and virtually no time is wasted during assembly.

Second, the language had to be powerful. This assembly language provides numerous resources for the programmer. In addition to the symbolic addressing capability of the existing assembler, LGPSAP, it provides for such things as address incrementing, self-reference addresses, equivalence statements, and "load-and-go" operation.

This paper will introduce you to the language, the operation, and some of the applications of this assembly program.

2. THE ASSEMBLY LANGUAGE

The language of this system is based largely on the LGPSAP language. For discussion we can divide it into three main components: instructions, constants, and assembly-directing commands.

Instructions

An instruction is made up of two basic parts, the operation field and the operand field.

Operation field. The operation field consists of one to four characters, all of which are assembled into bits zero through fifteen of the assembled instruction. All LGP-30 operations can be represented along with those of systems such as 24.1 & 24.2 .

Examples of possible operation fields:

h' 800t' 2n' 803p' j00h'

Constants

Two types of constants may be assembled by this system, hexadecimal and alphanumeric. A constant must be labeled, and the type of constant is denoted by the last letter of the label.

Hexadecimal constants. A label ending in "x" indicates that the next two fields each contain four characters of a hexadecimal constant.

Examples:

Code:	Assembled as:
'consx'5j21'kwgf'	5j21kwgf
'la30x''2'	00000002
'talyx'j001''	j0010000

Alphanumeric Constants. A label ending in "z" indicates that the next field is an alphanumeric constant which is to be stored in six bit form.

Examples:

Code:	Assembled as:
'endz'end'	0004fjff

Assembly-Directing Commands

Assembly can be directed by the programmer in two ways: by pseudo-operations in the operation field, and by special flags in the label field.

There are four pseudo-operations: START, REFER, BLOCK, and EQUIV.

START specifies the location at which instruction storage is to begin in memory, and REFER specifies the reference or base address of the program.

Examples:

```
start'2200'refer'200'
b'/'+'1'
```

The instruction would be stored at location 2200 and would be assembled as b2201.

```
start'2200'refer'0'
b'/'+'1'
```

The instruction would be stored at location 2200 and would be assembled as b0001.

BLOCK causes the assembler to skip over the number of locations specified (in track and sector form) by the numeric operand.

Examples:

'stor'block'20'	The next instruction will be assembled in location stor + 20.
block'108'	Skips over 64 + 8 = 72 locations.

EQUIV defines the location label by assigning it the same absolute equivalent as the operand. The operand may be an absolute address or a previously defined symbolic address.

Examples:

```
'here'equiv'2400'
'there'equiv'here'
```

This sequence first defines "here" as the address 2400, and then defines "there" as the same address.

The two special flags which may appear in the label field are WAIT and END.

WAIT (coded as 'wait') causes the assembler to stop and then proceed when the start switch is pressed. This command allows segmentation of tapes, combining of separately punched programs, etc.

END indicates the end of the source program. It is followed by a transfer address which may be blank, an absolute address, or a symbolic address. After a breakpoint stop, control is transferred as directed by this transfer address. A blank transfer address transfers to the beginning of the assembler to assemble another program. An absolute or symbolic address transfers control to that address. If the address is symbolic, it must have been defined in the program just assembled.

Assembly-Directing Commands - (continued)

Examples:

Code:	Action:
'end''	transfers to beginning of assembler.
'end'1200'	transfers to location 1200.
'end'here'	starts assembled program at "here."

3. OPERATION OF THE ASSEMBLY PROGRAM

The assembler itself occupies tracks 32 through 39, and uses tracks 40 through 63 for symbol table and working storage. Tracks 3 through 31 are available for assembled program storage.

Starting the assembly. After loading the assembler hex tape, place your source tape in the reader, with the six-bit switch down, and press the start switch.

The assembly process. To process symbolic addresses, the symbol table scheme of LGPSAP is used. The symbolic address, regarded as a number, is used to compute an address in the symbol table area. If the computed location is occupied by a symbol other than the one being processed, successively higher locations are tried until an unoccupied location is found or until a symbol table entry matches the symbol being processed. The LGPSAP algorithm performs far better than others tested.

The symbol table has associated with it a symbol address table which contains the absolute equivalents of the defined symbols in the symbol table. If a symbol has been used as an operand but is not yet defined, it appears in the symbol table, but its entry in the symbol address table is flagged with a sign bit. A threaded list is linked to this flagged entry, each element of this list containing a storage location where the address needs to be filled in when the symbol is defined. When the symbol becomes defined, the locations whose addresses are stored in the threaded list are filled in with the absolute address. It is this procedure which makes one pass assembly possible.

Error flags during assembly.

There are six error flags which may be printed out during assembly. They are printed in red so they will stand out from the source program.

APPLICATIONS - (continued)

it possible to use the computer to evaluate students' programs by means of a "grader" routine. The student ends his program with a 'wait' , and the operator follows the student's tape with a tape (prepared by the instructor) of equivalence statements which fill in the source data addresses and subroutine calls in the student's program. An 'end' statement then transfers control to the grader routine which supplies data to the student's program and evaluates his results.

In an engineering or scientific installation, this assembler will make it possible for you to get a short problem on the computer quickly for an "express run". For even a short program (one or two tracks) it will usually take less time for the assembler to fill in the addresses than for you to do it.

The symbolic program and the input data may be punched on the same tape. At the end of assembly the operator lifts the six-bit switch, presses the start switch, and the program commences running, using the data from the tape.

By reducing the size of the symbol table and modifying a few addresses, it would be possible to put this assembler on the drum along with 24.2 and still leave enough room to assemble an eight track program. Thus it would be possible to write a program in "symbolic 24.2" and have it running with a minimum expenditure of time and effort.

5. CONCLUSION

In this brief introduction to a one-pass assembly program, we have looked at its language, its operation, and some of its possible applications. The advantages of this assembler -- speed, convenience, and flexibility -- are available now. The program is running and checked out. I hope you will find that it "pays its way" and helps you to get even more of your money's worth from your computer system.

Supplement to "A One-Pass Assembly Program for the LGP-30"

By T. D. Lassagne

This supplement contains additional information on the assembly language for the information of programmers. This information will be contained in the program writeup when it is issued.

Operation Field. The operation field may not be left blank unless the instruction has a location label. A "z" should be punched in the operation field rather than leaving it blank. There is one exception to this general rule; this is the case where both the operation field and the operand field are blank. This case is assembled as z0000 .

Example:

p'1600'''	is assembled as	p1600 z0000
p'1600''3200'	would cause improper operation of the assembler. It should be written p'1600'z'3200'	

Operand Field. If an absolute address is used in the operand field, the instruction letter does not need to have an "x" before it as in LGPSAP. If the "x" is present it will be stored on the drum as part of the assembled instruction.

Alphanumeric Constants. The characters in an alphanumeric constant include only those characters which are normally read into the accumulator from the tape. This does not include the carriage return, code delete, etc.

START & REFER. If the START and REFER statements are missing from the beginning of a program, the assembler assumes that you meant to write start'0300'refer'0300' . If just one of the statements is missing, it is assumed to have the address 0300 . (In the 24.2 version, the assumed addresses would be 4000 .)

BLOCK. The locations skipped over by the BLOCK statement are not changed in any way, nor is anything stored in them. If you desire zeroes in the skipped over locations, the drum clear routine should be used prior to loading the assembler.

EQUIV. The operand of an EQUIV pseudo-op may not have an increment. Further, the location label of an EQUIV statement may not be a label ending in "x" or "z". If the operand is a symbol which has not yet been defined, the assembler will print an "ad" error flag and the EQUIV statement will be disregarded.

WAIT. The 'wait' stop is a breakpoint 8 stop.

END. The program stop after assembly is a breakpoint 4 stop. In the case of a blank transfer address, there is an unconditional stop between the breakpoint 4 stop and the transfer to the beginning of the assembler. That is, the Start button must be pressed twice to start assembly again if breakpoint 4 is up. If the transfer address is an undefined symbol, the assembler prints an "ad" error flag and treats the transfer address as if it were blank. One note: the transfer address should normally be blank if the START and REFER addresses are not the same; in this situation the program is not in the proper location for running.

A Programming Example. This short example is a program to input and store 20 words in successive locations starting at "stor" . It demonstrates a few of the features of this assembler.

The first column shows the source code, the second the storage location of the assembled instruction, and the third the assembled instruction itself.

```
start'2000'refer'5000'
b'aster'                    2000        b5012
y'/'+'4'                    2001        y5005
c'dump'                     2002        c5014
p''                         2003        p0000
i''                         2004        i0000
h''                         2005        h0000
b'/'-'1'                    2006        b5005
a'la29'                     2007        a5011
y'/'-'3'                    2008        y5005
s'test'                     2009        s5013
t'/'-'8'                    2010        t5002
'la29'z'1'                  2011        z0001
'astor'z'stor'              2012        z5015
'test'h'stor'+'20'          2013        h5035
'dump'''                    2014        z0000
'stor'block'20'
'end''
```

Before execution, this program would be moved to location 5000.

Source Tape Punching. Because this assembler does not print an assembly listing, the source tape need not be punched one instruction to a line, but may be punched in straight line form. For example, the programming example above would be punched as follows:

```
start'2000'refer'5000'
b'astor'y'/'+'4'c'dump'p''i''h''b'/'-'1'a'la29'y'/'-'3'
s'test't'/'-'8''la29'z'1''astor'z'stor''test'h'stor'+'20'
'dump''''stor'block'20''end''
```

Operation procedure.

To assemble a program:

1. Load the assembler and direct a "stop and transfer" to location 3200 (4800 in the 24.2 version). This stop and transfer command is normally on the end of the hex load tape.

2. Put the source tape in the reader and put the six-bit switch down.

3. Push START.

The assembler will:

1. Type "scoop". (This is the current name of the assembler).

2. Initialize the symbol tables (taking about 1 1/4 minutes).

3. Commence reading the tape and assembling.

When the END statement has been read, the assembler will:

1. Search the symbol table for undefined symbols.

2. List the undefined symbols under the heading "undef--".

3. Print the last location occupied by the assembled program.

4. Stop (breakpoint 4) and transfer.

Punched Comments. Generally, comments should stay on the coding sheet and not be punched on the source tape. Special precautions have to be taken to avoid trouble if they are punched on the tape. If you are reading the tape with a photoreader, leave the comments off entirely unless it has a modified input circuit.

Punched Comments. - (continued)

If reading with the flexowriter, comments may be included with the following precautions:

1. If the next line of code had a location label, follow the comment with a tab.

2. If the next line of code does not have a label, follow the comment with a tab and the n (4-n) spaces, where the operation field on the next line had n characters. Failure to take this precaution will result in some of the comment characters appearing in a spurious "op" error flag being printed after the operation field. All in all, it's safer to dispense with comments on the punched tape.

Questions, Corrections, etc. I will be happy to answer any questions you have on any aspect of the assembler and will gratefully receive any corrections or suggestions you care to bring to my attention. I intend to work on this assembler some more and add more features. Let me know what you can use. Here is my address:

Ted Lassagne
Box 7365
Stanford, California

GE Modifications to LGP SAP (Scoop)

LGP-30 CODING SHEET

PREPARED FOR:				PAGE 1 OF
JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY:	PROGRAM CHECKED BY:	DATE
PROBLEM:				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CODE RET	CONTENTS OF ADDRESS	NOTES
	,				,			G-E hex input, tally routine @ 0000
	,	☒						
HEX INPUT		0000	C	0014	,			
		04	80	I 0000	,			
		08	H	0014	,			word 2
		0J	C	008J	,			word 1
		10	80	I 0000	,			enter hex word
		14	[hh	h c L L LL]	,			execute code wd
		18	B	008J	,			word 2
		1J	S	003J	,		0004 words	incr + decr.
		20	U	002J	,			
~~ALTER~~		24	~~A~~	~~2754~~	,			"refer" +6131)
		28	~~U~~	~~024~~	,			
		2J	T	0000	,			
		30	H	0014	,			word 1
		34	C	008J	,			word 2
		38	U	0010	,			
		3J	000	WWWWJ	,			
PRINT B →		40	80	P 0200	,			C1
SEQ.		44	80	I 3000	,			
		48	80	P 0200	,			C2
		4J	80	I 3000	,			
		50	80	P 0200	,			C3
		54	80	I 3000	,			
U02K4		58	~~80~~	~~P 0200~~	,			C4 command
		5J	80	I 3000	,			
		60	80	P 0200	,			C5
		64	80	I 3000	,			
		68	80	P 0200	,			C6
		6J	80	I 3000	,			
		70	80	P 0200	,			C7
		74	80	I 3000	,			
		78	80	P 0200	,			C8
		7J	80	I 3000	,			

☒ CARRIAGE RETURN

LGP-21 CODING SHEET

PREPARED FOR:				PAGE / OF
JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY	PROGRAM CHECKED BY:	DATE
PROBLEM:				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CGE RET	CONTENTS OF ADDRESS	NOTES
	,							
	,	☒						
		00 80	~~4~~ 0	B 0 0 K 8	,			
		84	P	0 2 0 0	,			,
		88	U	0 0 0 0	,			exit
		8 J	~~~~~~		,			word 2
PRINT SIGNIF		90	T	0 0 4 0	,			NEG, ALL PRT
ONLY IN Acc		94	S	0 1 0 8	,	✓	0010 0000	C_1 C_2 C_3
R0088		98	T	0 0 K 4	,			no C_1 C_2 C_3 check C4
U 0000		9 J	S	0 1 6 4	,	✓	0000 0000	+0010 0000 −1000 0000 +0010 0000
		F 0	T	0 0 J 0	,			no C_1 check C_2 C_3
		F 4	A	0 2 W J	,	✓	1000 0000	starts C_1 ✓
		F 8	U	0 0 4 0	,			
		F J	A	0 1 0 4	,		0100 0000	
		G 0	8 0 0 I	3 0 0 0	,			
		G 4	U	0 0 4 J	,			shift + print 6
		G 8	~~~~~~		,			dump
		G J	b	0 0 0	,			b'
		J 0	A	0 1 8 8	,		0100 0000	check C_2 C_3
		J 4	T	0 0 F J	,			no C_2 starts w/C3 ✓ shift 6
		J 8	A	0 1 9 0	,		0100 0000	starts C_2 ✓ shift 6
		J J	U	0 0 4 4	,			shift print 7
		K 0	W W W W W W W J		,			
		K 4	8 0 0 I	3 0 0 0	,			check C4
		K 8	8 0 0 I	3 0 0 0	,			C4 to C_2 pos.
		K J	A	0 1 8 8	,		0100 0000	+1000 0000 −0100 0000 check C_2 pos.
		Q 0	T	0 0 8 J	,			no C4, check C5
		Q 4	A	0 1 9 0	,		0100 0000	starts w C4 ✓
		Q 8	U	0 0 5 4	,			shift + print 5
		Q J	8 0 0 I	3 0 0 0	,			check C5
		W 0	A	0 1 8 0	,		0100 0000	
		W 4	T	0 1 1 0	,			check C6
		W 8	A	0 1 F 4	,		0100 0000	starts C5 ✓
		W J	U	0 0 5 J	,			shift + print 4

LGP-30 CODING SHEET

PREPARED FOR:				PAGE 1 OF
JCG NO.	PROGRAM NO.	PROGRAM PREPARED BY:	PROGRAM CHECKED BY:	DATE
PROBLEM:				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CODE RET	CONTENTS OF ADDRESS	NOTES
	,				,			
	,	☒						
		0100	01000000		,			
		04	01000000		,			
		08	00100000		,			
		0J	~~~~~~		,			"start"
→		10	80	0I3Q00	,			check C6
		14	A	0188	,		0W00 0000	
		18	T	012J	,			no C6 check C7
		1J	A	0100	,		0W00 0000	starts C6
		20	U	0064	,			shift + print C6 C7 C8
		24	E	00K0	,		WWWW WWWJ	
		28	U	02G8	,			
		2J	80	0I3Q00	,			
		30	A	0188	,		0W00 0000	
		34	T	0148	,			no C7, check C8
		38	A	0100	,		0W00 0000	starts C7
		3J	U	006J	,			shift + print C7 C8
print out program		40	U	028J	,			
		44		3WWJ	,			
		48	80	0I3Q00	,			
		4J	A	0188	,		0W00 0000	
		50	T	0080	,			no C8, stop code only
		54	A	0100	,		0W00 0000	print C8 only + shift
		58	U	0074	,			
221J →		5J	H	010J	,			start
		60	U	2068	,			
		64	0W	W0 0000	,			
		68	W	000 0000	,			
		6J	000	WWWWJ	,			
		70	~~~~~~	FINAL	,			
		74	~~~~~~	INITIAL	,			
		78	WW	WW 0000	,			
		7J			,			

☒ CARRIAGE RETURN

LGP-21 CODING SHEET

PREPARED FOR:				PAGE / OF
JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY	PROGRAM CHECKED BY:	DATE
PROBLEM:				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CGE RET	CONTENTS OF ADDRESS	NOTES
	,							
	,	☒						
		0180	OWD00000		,			
		84			,			
		88	OWO00000		,			
		8J	WWWWWWWQ		,			
		90	0100000		,			
		94	000J0000		,			
		98	8000		,			
		9J			,			
		F0	000b0000		,			
		F4	0100000		,			
PRINT FROM		F8	B017C		,			FINAL
INITIAL TP		FJ	S0174		,			INITIAL @ bit 30
FINAL WITH		G0	I3Q00		,			to bit 12
LOADING CODE		G4	I3Q00		,			18 bits
(INITIAL) 0174 (FINAL) 0170 SET		G8	~~I~~ I3Q00		,			
		GJ	400A01F0		,		000 b 0000	
R0248		J0	A0174		,			INITIAL
U01F8		J4	H01QC		,			b
		J8	H00GJ		,			b'
		JJ	A0194		,		000 1 0000	c[I
		K0	R0088		,			Print 8
		K4	U0040		,			
		K8	B016J		,		4000 0000	C. Return
		KJ	P0200		,			
		Q0	hhhb0000		,			bring word
		Q4	R0088		,			
		Q8	U0090		,			print word
		QJ	U01W4		,			
		W0			,			
		W4	b00GJ		,			b'
		W8	S016J		,		W WWWJ	-1@hhh, + 04@ ddd
		WJ	h01Q0		,			b

hhh c xxx

LGP-30 CODING SHEET

PREPARED FOR:				PAGE 1 OF
JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY:	PROGRAM CHECKED BY:	DATE
PROBLEM:				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CODE RET	CONTENTS OF ADDRESS	NOTES
	,				,			
	,	☒						
		20.0	T	0240	,			if negative, exit, print C.R. only
		04	U	0210	,			if pos, hold in b'
		08	4	00000000	,			
		0J		2	,			
		10	h	00EJ	,			b'
		14	E	02W8	,		WJ	sector bits
		18	400	S0200	,		02	
		1J	T	0234	,			need 2 cr
		20	A	02QP	,		02	
		24	E	02QJ	,		1J	
		28	S	02QJ	,		02	
		2J	T	01K8	,			need 1 cr
		30	U	01Q0	,			next word
		34	B	0218	,			C.R.
		38	P	0200	,			
		3J	U	01KJ	,			
		40	B	0208	,			C.R.
		44	P	0200	,			
		48	U	0000	,			
ENTER INITIAL +		4J	INIT + FINAL		,			
FINAL, PRINT →		50	B	018J	,		WWWW WWWQ	
(GE HEX PCH)		54	8	00I02000	,			
		58	H	024J	,			I+F
		5J	T	0280	,			ONLY I entered
		60	E	0144	,		3WWJ	FINAL BITS
		64	H	0170	,			FINAL
		68	B	024J	,			I+F
		6J	M	0198	,		8000	→16
		70	H	0174	,			INITIAL
		74	R	0248	,			PRINT I to F
		78	U	01F8	,			
		7J	U	0250	,			

☒ CARRIAGE RETURN

LGP-21 CODING SHEET

PREPARED FOR:

PAGE 1 OF

JOB NO. | PROGRAM NO. | PROGRAM PREPARED BY: | PROGRAM CHECKED BY: | DATE

PROBLEM: | TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CGE RET	CONTENTS OF ADDRESS	NOTES
	,							
	,	☒						
		2 8 0	E 0 2 W 4		,		3 W W J	INITIAL ONLY
		8 4	H 0 1 7 0		,			= FINAL
		8 8	U 0 2 7 0		,			HOLD INITIAL ALSO
PRINT →		8 J	B 2 5 F 8		,			"FINAL"
LISTING TO		9 0	~~S 2 3 9 0~~		,		S 0 1 0 J	"START" @30
LOAD		9 4	I 3 Q 0 0		,			
STARTING		9 8	I 3 Q 0 0		,			
AT		9 J	~~8 0 0~~ I 3 Q 0 0		,		~~0 0 0 1 0 0 0 0~~	← 18 @12
"REFER"		F 0	0 4 0 A ~~2 3 5 0~~		,		~~W W W W~~ (START)	= hhh x start
UNLOAD FROM A010 J		F 4	~~S 0 7 6 J~~		,		~~0 0 0 1 0 0 0 0~~	"START" = hhh bstart
"START"		F 8	H 0 1 Q 0		,			b
TO FINAL		F J	H 0 0 G J		,			b'
~~H 0 0 2 4~~		G 0	~~A 2 7 5 4~~		,		W W W W 0 0 0 0	
~~0 0 0 2 4~~ A W F 0		G 4	~~A 2 7 5 4~~		,			"REFER"
		G 8	R 0 2 4 8		,			
		G J	U 0 1 J J		,			add c, print
		J 0	8 0 0 B 0 2 J 0		,			
		J 4	P 0 2 0 0		,			
		J 8	P 0 2 0 0		,			"
		J J	Z 0 0 0 0		,			
		K 0	U 2 0 0 0		,			another assembly ?
H 0 0 G 8		K 4			,			temp
E 0 1 6 8		K 8			,			(W 0 0 0 0 0 0 0 0)
~~A~~ 0 2 F 0		K J			,			(0 4 0 0 0 0 0 0)
P 0 2 0 0		Q 0		~~X~~	,			
B 0 0 G 8		Q 4			,			temp
U 0 0 5 J		Q 8		~~X~~	,			
		Q J		1 J	,			
		W 0		2	,			
		W 4		3 W W J	,			
		W 8		W J	,			
		W J	1 0 0 0 0 0 0 0		,			

LGP-30 CODING SHEET

PREPARED FOR:				PAGE / OF
JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY	PROGRAM CHECKED BY:	DATE
PROBLEM:				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CODE RET	CONTENTS OF ADDRESS	NOTES
	,							
	,	☒						
		1Q80			,			
		84			,			
		88			,			
		8J			,			
		90			,			
		94			,			
		98			,			
		9J			,			
		F0			,			
		F4			,			
		F8			,			
		FJ			,			
		G0			,			
		G4			,			
		G8			,			
		GJ			,			
		J0			,			
		J4			,			
		J8			,			
		JJ			,			
		K0			,			
		K4			,			
		K8	H2754		,			
		KJ	E02W4		,			
		Q0	U22KJ		,			
2168		Q4	E2488		,		WWWWW	
		Q8	HLWF4		,			loop
		QJ	B276J		,			constructed word
		W0	E248J		,		WWWWW	
		W4	A1WF4		,			
		W8	H276J		,			
		WJ	U2066		,			

☒ CARRIAGE RETURN

LGP·30 CODING SHEET

PREPARED FOR:				PAGE / OF
JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY:	PROGRAM CHECKED BY:	DATE
PROBLEM:				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CODE RET	CONTENTS OF ADDRESS	NOTES
	,							
	,	☒						
2000→		1W00	400	2 1W00	,			
		04	P	0200	,			C. Ret.
		08	U	2008	,			
2460→		0J	400	B1W0J	,			
		10	P	0200	,			
		14	B	272J	,			
		18	U	246J	,			
2484→		1J	400	B1W1J	,			C. Return
		20	P	0200	,			
		24	B	25F8	,			
		28	S	2514	,			
		2J	H	25F8	,			
		30	U	2498	,			
24F8→		34	H	1WF4	,			Temp
		38	4J0	B1W38	,			(1) slash mark
		3J	P	0200	,			
		40	Z	0400	,		8S 4	
		44	Z	0000	,		ZON	
		48	B	1WF4	,			temp
		4J	U	24G4	,			
2510→		50	400	B1W50	,			
		54	P	0200	,			
		58	U	2478	,			
2690→		5J	N	1W94	,		4 0000	C1@24, shift 18 b.
		60	P	0200	,			
		64	B	269J	,			C2@24
		68	N	1W94	,		4 0000	
		6J	P	0200	,			
		70	B	25F8	,			
		74	N	26F8	,			
		78	U	26FJ	,			
26JJ→		7J	N	1W94	,		4 0000	char @24

☒ CARRIAGE RETURN

LGP-21 CODING SHEET

PREPARED FOR:				PAGE / OF
JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY	PROGRAM CHECKED BY:	DATE
PROBLEM:				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CGE RET	CONTENTS OF ADDRESS	NOTES
	,							
4C 1W80	,	☒						
		1W80	P	0200	,			
		84	U	26K4	,			
26J0 →		88	0J	0B1W88	,			SP
		8J	P	0200	,			
		90	U	26K4	,			
		94	4	0000	,			
22J8 →		98	H	1WF0	,			"Refer"
		9J	U	2068	,			
		F0			,			Refer
		F4			,			temp
22F4 →		F8	Z	0800	,			"Wait" label
		FJ	Z	0000	,	ON OFF		
		G0	U	242J	,			
25J →		G4	H	1WF4	,			original
		G8	E	1WK8	,		06186186	5,6 bits in char
		GJ	C	00G8	,			5,6 pattern
		J0	S	00G8	,			5,6 pattern
		J4	A	1WK1	,		02082080	± 1 02 to change all 6's to 5's
		J8	E	1WQ0	,		79279278	subroutine exit only
		JJ	E	1WF2	,			original
		K0	A	00G8	,			5,6 pattern
		K4	U	2530	,			
		K8	06	18 61 86	,			
		KJ	02	08 20 80	,			
		Q0	79	27 92 78	,			
20SJ →		Q4	H	26WJ	,			
		Q8	E	0J44	,			Subtract
		QJ	H	0J0J	,			Subroutine
		W0	H	1WF0	,			
		W4	U	2060	,			
21J4 →		W8	E	02W4	,			
/ input		WJ	U	2168	,			

ASSEMBLER

000002wj

```
0gwc0000'
c0014'800i0000'h0014'c008j'800i0000''b008j's003j'
u002j'a2754'u0124't0000'h0014'c008j'u0010'swww j'
800p0200'800i3q00'800p0200'800i3q00'800p0200'800i3q00'u02k4'800i3q00'
800p0200'800i3q00'800p0200'800i3q00'800p0200'800i3q00'800p0200'800i3q00'
b00k8'p0200'u01qj''t0040's0108't00k4's0164'
t00j0'a02wj'u0040'a0104'800i3q00'u004j'f00ij000'90b00gj'
a0188't00fj'a0190'u0044'wwwswww j'800i3q00'800i3q00'a0188'
t00qj'a0190'u0054'800i3q00'a0180't0110'a01f4'u005j'

l0z0000'l0z0000'lz0000'400'800i3q00'a0188't012j'a0100'
u0064'e00k0'u02g8'800i3q00'a0188't0148'a0100'u006j'
u028j'3wwj'800i3q00'a0188't0080'a0100'u0074'h010j'
u2068'wwz0000'w00z0000'swww j'2wj''wwws0000''
w0z0000''w0z0000'wwwswwwq'l0z0000'h0000'8000''
b0000'l0z0000'b0170's0174'i3q00'i3q00'i3q00'400a01f0'
a0174'h01q0'h00gj'a0194'r0088'u0040'b01gj'p0200'
47b01q0'r0088'u0090'u01w4''b00gj's016j'h01q0'

t0240'u0210'400z0000'2'h00gj'e02w8'400s020j't0234'
a020j'e02qj's020j't01k8'u01q0'b0218'p0200'u01kj'
b0208'p0200'u027j'2wj'b018j'800i0200'h024j't0280'
e0144'h0170'b024j'm0198'h0174'r0248'u01f8'u0250'
e02w4'h0170'u0270'b25f8's010j'i3q00'i3q00'i3q00'
40a01f0'a010j'h01q0'h00gj'e0178'alwf0'r0248'u01jj'
800b02j0'p0200'p0200''u2000'h00g8'e0168'a02f0'
p0200'b00g8'u005j'lj'2'3wwj'wj'l00z0000'

2c1QKB ~~lqq427wj~~
h2754' e02w4' u22kJ'
246clqq4'
e2488'hlwf4'b276j'e248j'alwf4'h276j'u2068'

400blw00'p0200'u2008'400blw0j'p0200'b272j'u246j'400blwlj'
p0200'b25f8's2514'h25f8'u2498'hlwf4'4j0blw38'p0200'
400''blwf4'u24g4'400blw50'p0200'u2478'nlw94'
p0200'b269j'nlw94'p0200'b25f8'n26f8'u26fj'nlw94'
p0200'u26k4'j0blw88'p0200'u26k4'i0000'hlwf0'u2068'
300'800i3q00'800''u242j'hlwf4'elwk8'c00g8'
s00g8'alwkj'elwq0'elwf4'a00g8'u2530'61p6186'20p2080'
79qm9q78'h26wj'e0144'h010j'hlwf0'u2060'e02w4'u21g8'

ulw00'4f'b27k0'r26g8'u26g0'b27kj'r26g8'u26g0'
b27jj'h202j'a27w0'wwwh37wj't2024'b27wj'c203j'wwwc3wwj'
b203j'a27w0't2038'b26w8'h27q0'b2698'y25f8'ulwq4'
b20f0'y2594'r2530'u251j'h2738'e273j's27k4't2244'
s2748't20f0's27k4't2230's2758't20f0's27k4't2220'
r25j4'u2594'b2738's2774't20wj'b20f0'y2594'b2738'
s2788't20w0's27k4't21kj's2798't20w0's27k4't2200'
s27f8't20w0's27k4't2210'b26qj'r26g8'u26g0'b27j4'

y2594'b2738'r2590'u2568'n27k8'h276j'r2530'u251j'
h2738's27k4't21g4's27w4't213j's27k4't21j0'b2738'
r2650'u262j't2150'u21g8'a27k4't2168'b25w4'y2164'
b2738'c2g00'h27q4'b27q0'y21g0'y2184'y2190's2514'
t21j8'h3004'y27q0'h27q4'h3004'b27g0'a2728'y21f8'
```

b26wj'ulww8'b26w0'r26g8'u26g0''u2000'r2564'
u2554'h2758'a26wj'y26wj'b2518'a2758'y2518'u2660'

r2564'u2534'y26wj'ulw98'r2564'u2534'y25f8'u015j'
r2564'u2534'a276j'u21g8'r2564'u2534'c2738's2738'
u2228'r25j4'u2594'b27j4'y2594'r2530'u251j'h2738'
e273j's27k4't2360's27k4't23jj's2004't2284's27k4'
u2254'b2738's2750't22f8's27k4't2430's2760't22f8'
s27k4'tlwf8'r23j0'u2368'r2530'u251j'h2738's2784'
t22jj's27k4't232j'b2794'y22q8'b26wj'h2754'h3g00'
b276j't22qj'u20f8'm27g4'y22w8'y230j'b3004'h27q4'

y231j'y2320'b27q0'h3004'b230j'y27q0'b2754'a0408'
y0408'b27q4'u22q4'b2764'y22q8'r2530'u251j'h2738'
r2650'u262j't2350'u22k8'b270j'r26g8'u26g0'u2060'
c276j'u2418'r2628'u25k0't23f4'a2728'y22kj'y2380'
b3g00't23gj'b2700'r26g8'u26g0'r2530'u251j'h2738'
u22q8'b25w4'y23g8'a2720'y22kj'b2738'h2fw8'h276j'
u22g0'b278j'u23k0'b2778'y22q8'r23j0'u2368'r2530'
u251j'h2738'u22k4'b2738'r2590'u2568'n27k8'c276j'

r2530'u251j'r2590'u2568'm2768'a276j'h276j'u2068'
b2738'u2418'800'u2060'b2708'y24jj'r2530'u251j'
h2738's27k4't24j4'a27k4'r2650'u262j't24g8'y24g4'
ulw0j'b272j''h2470'wwwb3wwj't24jj'b2470'a27w0'
t246j'ulwlj'wwwq'wwws0000'u2068''i3q00'i3q00'
r0088'u005j'ulw34''400'u0140'b270j'r26g8'
u26g0'b271j'u245j'u24k0'b271j'y24g4'b2740'r26g8'
u26g0'b275j'r26g8'u26g0'b2790'y24jj'b2470's2720'

y2504'b285j'r26g8'u26g0'ulw50'4'u2478'c27q4'
i0200'n27k4'e27gj'ulwg4'u2440'r2530'u251j'800r2590'
u2568'm254j'u2564'200z0000'a27q4'h27q4'e2704'm2724'
a27q4'u2660'e2730'm2734'h27q4'e27q8'm2724'a27q4'
h27q4'e274j'm2718'a27q4'u2544'u2598'b2744's25f8'
t25j8'b276j'h041j'b25f8'a2514'y25f8'b26wj'a2514'
y26wj'u224j'b27g8'u21jj'b2738'n279j'm27f0'e27f4'
a2770'e27fj'h27g0'a277j'y25w4'b2g00't2628's2738'

t260j's27k4't2620'b27g0'a27w0't25q4'b27j0'u21jj'
b27g0'e27f4'u2370'e26w4's27k4't2654'r2628'u25k0'
t2664'a2728'y264j'b3qq8'u2458'b2738'r2564'u253j'
u2650'c27q4's27k4'u2650'e27j8's27qj't2680'u2674'
n2514'a2780'y269j'n26f8'ulw5j'p0000'302'p2608'
b25f8'n26f8'40'u24f8'h27q4's27k4't2020's27w8'
tlw88'a2774'm27k8'ulw7j'p0000'b27q4'n26f8'e27gj'
u26g0'p0300'u26k4'l9lh2606'l95iq620'20p2082'3000'322'

l8quf606'rwj00'24k0'l9juf606'rj3j0'k00z0000'880z0000'2lk4'
l000'f00z0000'3800'q00b3800'lqm9q78'400z0000'lzj904'7q'
l94t2fjf'hlcs4'j'3sj000'ifjff'3lq'6'543pq800'
lwy7j2q'2060'200z0000'300'q00z0000'20z0000'2420'2800'
29w8'4gkd28gq'f3i7fqj'23qj'24w8'20f8'l0wpk7fj'5kylkgf'
400'7wj'604bk43q'wwwz07wj'q00z0300'y0000'l9lpf620'7wwswwwq'
l9qcf620'2598'3w00'j00h2800'86mgfj6'2'4000'470e8800'
3004'4l00'wz0w00'9wj'lz0004'24'lwswwwq'kwwc37wj'
''u2000''