TIMING AND OPTIMIZATION 8

Optimization is a programming technique which provides access to data and instructions with a minimum of nonproductive searching time. When a program is optimized for the LGP-21, the programmer utilizes the interlace arrangement of sectors around the disc in a manner which will be explained after instruction timing is fully understood.

TIMING

Generally, an instruction is said to be optimum if its four phases can be executed before the disc turns past the location of the next instruction in sequence. However, some instructions-such as multiply, divide, and input-require more than 18 word-times for their operations.

Since timing is an integral part of optimization, the 4-phase instruction cycle—already discussed in Chapter 3-is summarized here once more. Each phase is measured in computer word-times. During Phases 1 and 3 the computer searches for a specified sector and then activates the appropriate read/write head. This may take from one to several word-times. Phase 2 always requires one word-time; Phase 4 takes one word-time for all instructions except N-multiply, M-multiply, and Divide, which require 63, 65, and 66 word-times respectively.

The time required for the computer to read an instruction, execute it, and be ready to read the next instruction depends on whether an instruction is optimum or not. Figure 8.1 shows the various timing requirements:

Instruction	Optimum	Non-Optimum
Bring, Add, Subtract, Hold, Clear, Extract, Set Return Address, Store Address, and Shift	7.26 ms	58.11 ms
N-multiply, M-multiply, and Divide	58.11 ms	108.96 ms
Unconditional Transfer and Conditional Transfer	1.59 ms	Each sector beyond optimum adds .40 ms
Sense	7.26 or 14.52	

FIGURE 8.1 Optimum Timing Requirements

An instruction can be optimum only if its operand is located within a certain number of word-times. Figure 8.2 lists the range of optimum sectors for all instructions which can be optimized.

Instruction	Distance from Instruction Location to Optimum Operand in Word-Times
Bring: Add, Subtract, Hold, Clear, Extract, Set Return Address, and Store Address	2 through 16
N-multiply	2 through 81
M-multiply	2 through 79
Divide	2 through 78
Unconditional Transfer and Conditional Transfer	4 or more if transfer is active
Others	always optimum

FIGURE 8.2 Range of Optimum Sectors

Input Timing

An input instruction is held in Phase 3 of its cycle until a start signal is received from the input device. Therefore, the computer is not free to perform internal calculations during an input operation. The total time for executing any input instruction consists of three word-times plus the Phase 1 search and the time required for reading or typing.

Output Timing

Because of the buffering system built into the LGP-21, an output operation will not delay the computer unless the selected device is already in use. Thus, the computer is free to perform other internal calculations while the information is being output. During the time the output device is busy, the interlock for that device is turned ON. If that device is selected for output a second time while its interlock is ON, the computer will delay execution of the second print until the interlock is turned OFF. The interlock is turned OFF automatically when the device is free. If, during the time an output device is busy, a second instruction selects a different device for input or output, the second instruction will be executed without delay if that device is not busy. Therefore, the operation of two or more input/output devices can overlap in time. For example, if an output to the tape typewriter is followed by an output to the 151 Tape Punch, the computer will not delay on the second output instruction even though the typewriter is still busy. If the first and second output instructions both select the typewriter, the computer may delay on the second instruction until the typewriter is ready.

Print instructions must be 1/3 revolution apart for the 151 Tape Punch and not more than 2 revolutions apart for the 121 Tape Typewriter, if these devices are to operate at their rated speeds.

OPTIMIZATION

At the beginning of this manual, it was briefly mentioned that the physical characteristics of the memory disc are disregarded for general programming purposes as they vary from the 64 track/64 sectors concept used. Actually, the disc consists of 32 tracks with 128 sectors each. These sectors are not numbered sequentially within a track although the pattern of numbering is the same for all tracks. This system is based on an 18-word interlace pattern which positions consecutive words 18 sectors apart-a feature which aids in optimizing of instructions which are executed in sequence.

Fortunately, the programmer does not have to memorize this complex pattern in order to utilize optimizing for his programs. He can use the device illustrated in Figure 8.3, which will let him determine at a glance the optimum sectors for the operand of any instruction.

The device-called the Optimum Address Locator-shows the interlace pattern of the sectors on the memory disc. Each sector is represented by threedigits of which the second and third represent an actual LGP-21 sector number. The initial digit-which is either 0 or l-indicates whether the track used in conjunction with the sector is even- or odd-numbered. A 0 indicates that the sector is on an even-numbered track; a 1 an odd-numbered track.

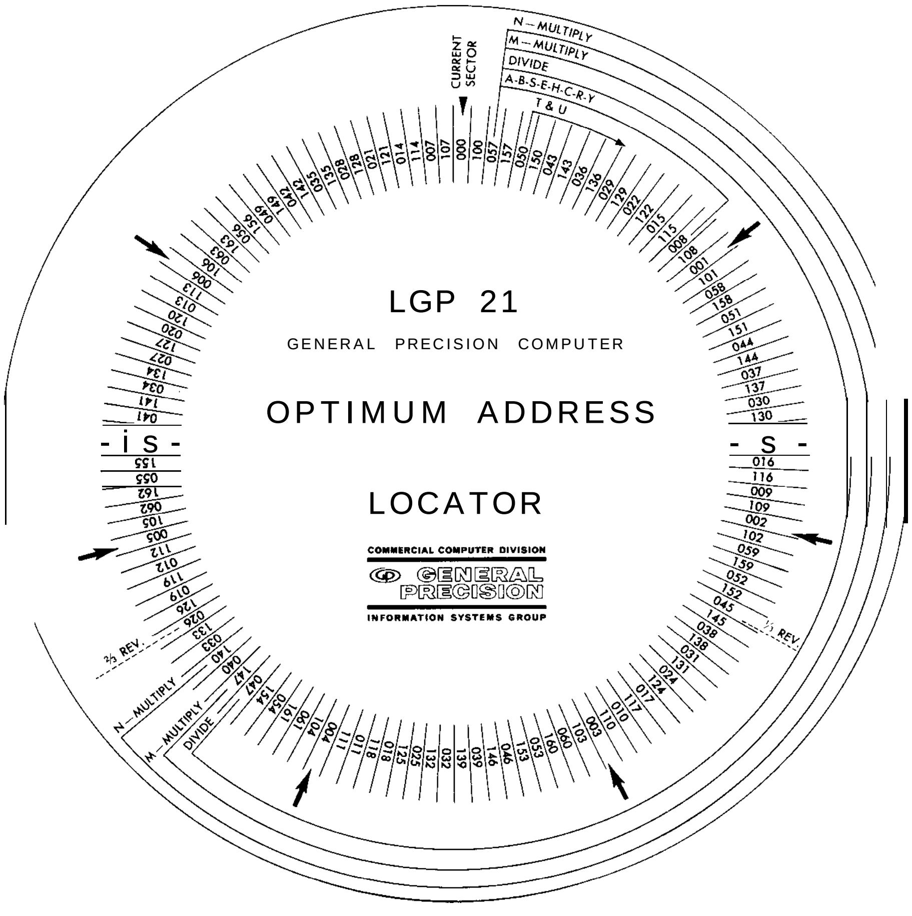

FIGURE 8.3 Optimum Address Locator

To determine the optimum operand address for an instruction with the help of the Optimum Address Locator, one must first locate on it the sector address of the instruction.

Assuming a Hold instruction is in location 1400, sector 00 must be found on the rotating center wheel of the device. Since track 14 is even-numbered, the sector address 00 must be preceded by another 0. The next step is to center 000 above the “Current Sector” indicator (on the larger, outside wheel). Adjacent to this indicator are five lines which delimit as many segments; each segment being headed by the name (or names) of the LGP-21 command to which it pertains. For example, the first segment applies to the T and U instructions; the fifth to N-Multiply.

The "H" for the Hold instruction can be found above the second segment. This indicates that the optimum operand sectors for that command are contained within the second segment area, namely between 057 and 008. In addition, all other sectors within this range also yield an optimum operand address for the Hold instruction above; namely, 057, 157, 050, 150, 043, 143, 036, 136, 029, 129, 022, 122, 015, 115 or 008.

If the Hold instruction had been in an odd-numbered location, say 1500, sector 00 on the center wheel would have to be preceded by a 1 to indicate the odd track number. With 100 centered underneath the "Current Sector" indicator, and the H command again located above the 2nd segment of the larger, outside wheel, the first possible optimum operand address for the instruction would have been 157, and the last 108. In addition, all sectors listed between 157 and 108 would be optimum for H1500.

At this point, one further general rule for using the Optimum Address Locator should be adopted as good practice: that is, to never use the outer limits of the optimum range for any instruction, if possible. The reasons for this rule will be explained later in this chapter, when a few additional concepts have been understood. Finally, the larger outside wheel of the Optimum Address Locator provides the sector location in which the next instruction to be executed will be found. This can be obtained from the black arrows which are placed, for clockwise reading, around the periphery of this disc. In other words, when the Current Sector indicator for H 1400 is placed on 000, the black arrows show that the next instruction would be located in 1401, etc.

Thus, by using the Optimum Address Locator, an operand address can be chosen which permits execution of the Hold instruction before the next instruction (in Location 1401) passes the read head. On the other hand,if an H1658 is in Location 1400, it is not an optimum instruction since sector 01 would have passed the read head before the computer could complete the execution of H1658; therefore, the disc would make a full revolution before Location 1401 could be read. Most instructions can be executed in approximately 1/7 of a disc revolution when operand addresses are optimum. This constitutes a significant saving in machine-time.

The U and T instructions are optimized somewhat differently than other instructions. If a U instruction is in sector 00 of an even-numbered track, the fastest transfer will be to sector 50 on an even or odd track; if the U instruction is in sector 00 of an odd-numbered track (e.g. 0100), the fastest transfer will be to Sector 50 on an odd-numbered track (e.g., 0150). Each subsequent sector will add one word-time to the minimum transfer time. The search for the sector of a T instruction occurs only during an actual transfer. If T1251 is given in Location 1700 and a transfer is not actually made, then the instruction in 1701 will be read as it passes the read head on that disc revolution.

For one more example of optimum programming, consider the following sequence of instructions.

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	'						
	'	☒					
		0000	B	1943	'		
		0001	A	2251	'		
		0002	C	1943	'		

The instructions B1943, A2251, and Cl943 must be brought into the Instruction Register before they can be executed. This can happen only when Locations 0000, 0001, and 0002 respectively are under the read head. If the read head is over sector 00 and is placing the instruction B1943 in the Instruction Register, then, if this instruction can be executed before the disc turns to sector 01, it will be an optimum instruction. Figure 8.3 shows that sector 43 for an odd-numbered track is between 00 and 01 and is within the optimum range for sector 00. The same logic applies to the next instruction, A2251 at Location 0001. Sector 51 for an even-numbered track lies between sector 01 and sector 02 and is within the optimum range for sector 01. Therefore, both these instructions are optimum. However, the next instruction, Cl943 in Location 0002, will not be optimum because sector 43 does not lie between sectors 02 and 03.

Two further concepts need to be introduced now, to explain why optimum addresses should not be selected on the extreme outer limits of the optimum range, whenever possible. The first pertains to the difference between relocatable and non-relocatable programs. A non-relocatable program is coded for storage in a particular area of memory and will not operate properly if stored anywhere else. A relocatable program can be stored anywhere in memory. It is normally written relative to Location 0000; that is, as if the first instruction will be stored in Location 0000. In actual operation, it can then be stored by a program input routine beginning with any location the programmer specifies. (The program input routine description explains how this is done.)

The other new concept pertains to address modification. As a relocatable program is being loaded, some operand addresses must be modified by the program input routine for proper operation of the program. However, *some* addresses—for example, those in Input and Print instructions-can not be changed because they represent standard selection codes for certain input or output devices. The same is true for sense Branch Switch instructions, since their address determines which Branch Switch is tested. Consequently, the program input routine must be informed of any addresses which are not to be modified. This is done by preceding such commands with an "X":

```
XZ0800'
80X10200'
XA2001'
XR1011'
XU1000
```

NOTE: The X is recognized by the program input routine, but it is not stored in memory.

Now it can be shown that, if a non-modifiable instruction has been written with an extreme outer-limit optimum operand address, and the program is relocated by an odd number of tracks, the instruction would no longer be optimum. For example, if the instruction

Location	Instruction
0200	XA0457

is part of a relocatable program, and if the program is relocated upward by three tracks, the instruction would appear as

Location	Instruction
0500	XA0457

and would not be optimum. For this reason, optimum addresses should not be chosen on the outer limits of the optimum range.

In closing, it might be mentioned that all LGP-21 subroutines programmed by the General Precision Commercial Computer Division have been optimized to effect savings in machine-time for the user. This is, however a more time-consuming effort than straightforward programming. Consequently, if a programmer decides to optimize a program, he should first compare the possible savings in machine-time with the added programming costs.