Chapter 16

The LGP-30 and LGP-21

The LGP-30 is a small computer with an Mp.drum. It is distinct from the first (and succeeding) generation computers using Mp.random_access and can be described by using the PMS diagram in Fig. 1. The LGP-21, a direct descendant of the LGP-30, having the same ISP, is also described by Fig. 1.

Since there is only one address/instruction, a method is needed for the optimal allocation of operands. Otherwise, each instruction might have to wait a complete drum (or disk) revolution each time a data reference is made. The LGP-30 provides for operand-location optimization by interlacing the logical addresses on the drum so that two adjacent addresses (e.g., 00 and 01) are separated by nine physical locations.[1] These spaces allow for operands to be located next to the instructions which use them. There are 64 tracks, each with 64 words (sectors). Each word is accessed by a track address of 6 bits and a word address of 6 bits. The sequence of words (sectors) within a track is 00, 57, 50, 43, 36, 29, 22, 15, 08, 01, 58, 51, 44, 37, . . . , 06, 63, 56, 49, 42, 35, 28, 21, 14, 07, 00. The time between two adjacent physical words is approximately 0.260 millisecond, and the time between two adjacent addresses is 9 × 0.260 or 2.340 milliseconds. The actual maximum t.access is 16.66 ms.[2]

Half of the instruction (15 bits) is unused. It could be used for extra instructions, indexing, indirect addressing, or a second (+1) address to locate the next instruction, all of which increase the preformance.

[1]The LGP-21 has a space of 18 words.
[2]The later LGP-21 appears to have a lower performance than the LGP-30 by about a factor of 3.

```
C ['LGP-30; technology: (113 vacuum tubes), (1350 diodes);
   power: 1500 watts; weight: 800 pounds; number produced:
   320 ~ 490; t.delivery: September 1956; descendant: 'LGP-21;
   Pc(1 address; 1 instruction/w; data: w,bv,i,fr; Mps(~ 2 w);
      operations: (+,-,×,/,∧,x 2))
   Mp(drum; t.cycle: 260 μs/w; t.access: (.260 ~ 16.6) ms;
      i.rate: 2.34 ms/w contiguous addresses:  4096 w; (31,1
      space) b/w)
   T(Flexowriter, paper tape)]

C ['LGP-21; technology: (460 transistors), (375 diodes); power:
   300 watts; weight: 90 pounds; number produced: ~ 150;
   t.delivery: December 1962;
   Mp(fixed head disk; cyclic; t.cycle: 400 μs/w; t.access:
      (0 ~ 52) ms; i.rate: 7.26 ms/w contiguous addresses:
      4096 w; (31,1 space) b/w)
   T(#1:32; Flexowriter, paper tape, analog, CRT, card)]
```

Fig. 1. LGP-30 and LGP-21 PMS diagrams.

The ISP, given in Appendix 1 of this chapter, is about the most straightforward in the book. There are only 16 instructions, and the program state is less than two words. Although the performance is limited because of an Mp.cyclic_access, an Mp.random_access would serve to make the ISP fairly similar to other faster computers, e.g., an IBM 701.

APPENDIX 1 LGP-30 AND LGP-21 ISP DESCRIPTION

Appendix 1

LGP-30 and LGP-21 ISP Description

```
Pc State
   A<0:30>                                   Accumulator
   C<18:23,24:29>                            Program Counter register
   Ov                                        Overflow, LGP-21 only on LGP-30 machine stops if an overflow
   Run

Pc Console State
   BP<4,8,16,32>                             Break Point switches
   TC                                        Transfer Control switch

Mp State
   M[0:77₈][0:77₈]<0:30>                     primary memory; 2^12 w; track and sector (word)

K State
 The following Input Output devices do not have synchronization description variables.  LGP-21 only.  LGP-30 has a Flexowriter.
   Input_device[0:31]<1:6>
      stop code                              condition signifying input device has read a special code
   Output_device[0:31]<1:6>

Instruction Format
   i<0:30>                                   instruction
      op<0:3>   := i<12:15>                  operation code
      t<0:5>    := i<18:23>                  track select bit on Mp
         t'<0:4>:= t<1:5>                    input-output select, LGP-21 only
      s<0:5>    := i<24:29>                  sector select bit of Mp
      skip condition := ((t<0:3> ∧ ¬ BP) ≠ 0)

Instruction Interpretation Process
   Run → (i ←M[C]; C ←C + 1; next            fetch
         Instruction_execution)              execute

Instruction Set and Instruction Execution Process
   Instruction_execution := (
   Z (:= op = 0) → (
     (t = 00000₂) → (Run ←0);                stop
     skip condition → (C ←C + 1);            sense BP and transfer
     i<0> → (Ov → (Ov ←0; C ←C + 1)));       sense overflow and transfer
   B (:= op = 1) → (A ←M[t][s]);             bring from memory
   Y (:= op = 2) → (M[t][s]<18:29>←A<18:29>); store address
   R (:= op = 3) → (M[t][s]<18:29> ←C + 1):  set return address
   I (:= op = 4) → (                         shifts, and input
      ¬ i<0> ∧ (t=62) → (A ←A x 2^6 {logical});
        i<0> ∧ (t=62) → (A ←A x 2^4 {logical}):
      ¬ i<0> ∧ (t≠62) → (input_6_bit):
        i<0> ∧ (t≠62) → (input_4_bit)):
```

APPENDIX 1 LGP-30 AND LGP-21 ISP DESCRIPTION (Continued)

```
        input␣6␣bit := (A ← A x 2^6 {logical}; next                    input processes
                        A<25:30> ← Input␣device[t']; next
                        (¬A<0> V stop code) → input␣6␣bit)              wait
        input␣4␣bit := (A ← A x 2^4 {logical}; next
                        A<27:30> ← Input device[t']<1:4>; next
                        (¬A<0> V stop code) → input␣4␣bit)
O (:= op = 5) → (0v,A ← round(A / M[t][s]));                           divide
N (:= op = 6) → (A ← A x M[t][s] {s.integer});                         multiply, save right
M (:= op = 7) → (A ← A x M[t][s] {s.fraction});                        multiply, save left
P (:= op = 10_8) → (
  ¬ i<0> → (Output␣device[t']<1:6> ← A<0:5>):                          print 6 bit
    i<0> → (Output␣device[t']<1:6> ← A<0:3>□1□0));                     print 4 bit
E (:= op = 11_8) → (A ← A ∧ M[t][s]);                                  extract
U (:= op = 12) → (C ← t□s);                                            unconditional transfer
T (:= op = 13) → (i<0> → ((A<0> V TC) → (C ← t□s));                    transfer control
                  ¬i<0> → (A<0> → (C ← t□s)));                         conditional transfer
H (:= op = 14) → (M[t][s] ← A);                                        hold and store
C (:= op = 15) → (M[t][s] ← A; next A ← 0);                            clear
A (:= op = 16) → (0v□A ← A + M[t][s]);                                 add
S (:= op = 17) → (0v□A ← A - M[t][s])                                  subtract
                )                                                      end Instruction␣execution
```