Hints Towards Good 7030 Programming

I. Generalities

1. Efficiency in computer problem solving involves the balancing of the following factors:
   a. Accuracy of results
   b. Analysis effort
   c. Programming time
   d. Debugging time
   e. Production run time
   f. Effectiveness in repeated use of program (possibly by a stranger)

   the relative ~~stress~~ weight of these factor varies from ~~program to program~~, problem to problem, and from ~~installation~~ individual to individual institution to institution. For small "one-shot" problems the trend is towards the emphasis on a, b, c, d.
2. Timing is important for much-traversed inner loops, but usually less important elsewhere.
3. There are usually many ways of doing the same problem.
4. The 7030 will not be efficiently used when the programmer tries to make it look like machine X (substitute in your ~~favorite~~ old acquaintance).
5. Advantages should be taken of special features in STRAP and MCP to minimize errors, and to simplify debugging.
6. Machine efficiency is gained by distributing the work over as many major units as possible, such that at any given time no major unit is ~~com~~ idle.
7. Memory conflict can be largely removed by putting instructions and data in separate memory box groupings.
~~8. The concurrent operation of the major units~~ autonomous
8. Information transmittal between autonomous major units is through buffer registers. The I-box buffers 1Y, 2Y and the Lookahead buffers levels LA0, LA1, LA2, LA3 should ~~not be, contain useful information~~ not be left empty over extended length of time. ~~Nor~~ Nor should they be constantly crowded by data with little information content.
9. It is perfectly permissible to use floating point operations on VFL quantities, binary operations on decimal quantities, as long as the answers are correct (Who can stop you, anyway?)

TCC Nov. 1960

II. Specifics

1. Floating point operations are usually E-box limited in timing (exceptions being L, LWF, DL, DLWF and ST). VFL operations involve extensive decoding and execution time, and are usually much slower that the floating point counterparts. I-box operations usually do not involve the E-box and the I-box time can be largely absorbed by floating point operations in the neighborhood.
2. I-box fetches are less efficient than E-box fetches, as the latter is greatly enhanced by Lookahead buffering. I-box fetches are made when the instruction is being decoded, and may wipe out buffered instructions in 1Y or 2Y.
3. SV, SC, SR are more time consuming than SX, for the latter does not call for I-box fetch
4. Information transfer from the E-box to the I-box is relatively time consuming, but is still faster than, say from the E-box to main memory, then immediately from main memory to the I-box.
5. Immediate operands require no fetch, and are to be preferred, particularly for I-box operations.
6. All VFL stores are fetch-and-stores.
   Every BB involve a fetch, VFL arithmetic, and a store
   Bind for non-index indicators is similar to BB except that a) I-box ~~operation~~ activity is less extensive and the fetch-store involve an internal operand ($IND).
7. VFL information will be processed more efficiently if ~~no~~ word boundary crossover is not present Otherwise there will be over-exercising of the memory and LA.
8. Take advantage of "forwarding", but avoid (if easily accomplished) other types of store-close-To-fetch
9. ~~Successive~~ Avoid consecutive stores, or they are time consuming. So is forwarding more than once.
10. I-box otherwise would be standing still and LA gradually drained.

10. All successful branch instructions will temporarily remove I-box buffer.
11. The following branches are considered unconditional by the I-box:
    CB and variants
    Bind for XF, XVLZ, XVZ, XVGZ, XCZ, XL, XE, XH
    and are performed correctly by the I-box.
12. The following are considered by I-box to be truly condition branches:
    Bind for non-index indicators
    BB
    I-box makes the tentative assumption that the branch is ~~pas~~ not successful, and processes ahead. If the assumption proved wrong, "branch recovery" will be performed, which requires the cleaning of I-box, restoring of pre-processed index registers, cleaning of LA before resumption of normal activities. Conditional branches should be largely unsuccessful, if even if an additional (unconditional) branch instruction has to be added to the program.
13. BD, RNX, T and SWAP requires cleaning of LA.
14. Interruption involves cleaning of I-box, restoring of index registers, execution of a pseudo Bind instruction, fetching and execution of a free instruction before the resumption of normal activities. Judicious use of this feature, however, allows the writing of inner loops with few long ~~I-box~~ branch instructions.
15. VFL instructions require from 2 to 6 levels of LA and ~~are slow~~ involve the slower byte processing. The power of such instructions (particularly the logical connective instructions), however, frequently compensates for the slow speed.
16. Begin a loop with a full-word address, even if a CNOP has to be placed just prior to the loop (why fetch for something one doesn't use?).
17. For optimum speed, fetch-type I-box instructions should occupy the <u>second</u> half of a full word, to avoid wiping of a useful Y buffer level.
18. The following special registers are bona fide memory locations, and are subject to the usual memory restrictions:
    0. ($Z), 4. ($MB), 13. ($RM), 14. ($FT), 15. ($TR).
    The load factor instruction thus involves ~ fetch and a store.

T.C.C. Nov. 1960