A U T O N E T I C S
A DIVISION OF NORTH AMERICAN AVIATION, INC.
INDUSTRIAL PRODUCTS
3584 Wilshire Blvd., Los Angeles 5, Calif.

June 1, 1960

RECOMP TECHNICAL BULLETIN NO. 9

TITLE: RECOMP OPTIMIZATION TABLES AND RELATED TIMING CONSIDERATIONS

PURPOSE: To present to the programmer tables enabling him to code optimally, and to utilize the so called "Gray Area" of memory.

EFFECTIVE DATE: June 1, 1960

CONTENTS: The following pages contain two tables which will be useful to the programmer who is interested in coding optimally, that is, in minimizing the total running time of a given program. In addition, there is a discussion of the general timing considerations involved when coding RECOMP II in machine language.

TABLE I

RECOMP II time factors table. (All numbers octal)

Instruction		k	i ***
ADD		02	02
ALS		(n)	(02 + n)*
ARS		(n)	(02 + n)*
CFL		42	47
CFV		42	47
CLA		02	01
CLS		02	01
CTL		02	07
CTV		02	07
DIS	(Command)	02.0	42
	(B C D)	02.1	(04 + 2m) Max: 42
DIV		02	51
DSL		02	51
DSR		02	52
DVR		02	52
EXT		02	01
FAD		02	(05 + d + n) Max: 54**
FCA		02	01
FCS		02	01
FDV		02	60**
FMP		02	57**
FNM		(h)	(02 + n)*
FSB		02	(05 + d + n) Max: 54**
FSQ		02	54
FST		42	41
MPR		02	50
MPY		02	50
PNC		(j)	(j)
PNW		02	(j)
PTC		(j)	(j)
PTW		02	(j)
RDY		(j)	(j)
RDZ		(j)	(j)
SAX		43	40
SQR		02	52
STA	(Main memory)	02	40
	(L or V loop)	06	00 Max: 15*
STØ		43	40
SUB		02	02
TMI		05	03*
TØV		05	03*
TPL		05	03*
TRA		05	00*
TSB		05	02*
TSC		05	02*
TSD		05	02*
TYC		(j)	(j)
TYW		02	(j)
TZE		05	03*
XAR		(h)	02*

For explanation of symbols, (*,**,***,d,h,i,j,k,m,n) see following page.

TABLE II

Optimum address for command pair located in word 0000, where right hand operation is TRA.

OPR	ADDRESS OF OPERAND	TRA	ADDRESS OF NEXT INSTRUCTION
ADD	0002	TRA	0011
ALS	00(n)	TRA	00(7+n)
ARS	00(n)	TRA	00(7+n)
CFL	0042	TRA	0016
CFV	0042	TRA	0016
CLA	0002	TRA	0010
CLS	0002	TRA	0010
CTL	0002	TRA	0016
CTV	0002	TRA	0016
DIS	0002.0	TRA	0051
DIS	0002.1	TRA	00(13+2m)
DIV	0002	TRA	0060
DSL	0002	TRA	0060
DSR	0002	TRA	0061
DVR	0002	TRA	0061
EXT	0002	TRA	0010
FAD	0002	TRA	0063*
FCA	0002	TRA	0010
FCS	0002	TRA	0010
FDV	0002	TRA	0067*
FMP	0002	TRA	0066*
FNM	——	TRA	0055
FSB	0002	TRA	0063*
FSQ	0002	TRA	0063
FST	0042	TRA	0010
MPR	0002	TRA	0057
MPY	0002	TRA	0057
SAX	0043	TRA	0010
SQR	0002	TRA	0061
STA	0002	TRA	0047
STA	7766	TRA	0013
STØ	0043	TRA	0010
SUB	0002	TRA	0011
TMI	0005	TRA	0010
TØV	0005	TRA	0010
TPL	0005	TRA	0010
TZE	0005	TRA	0010
TSB	0005	TRA	0007
TSC	0005	TRA	0007
TSD	0005	TRA	0007
XAR	——	TRA	0007

* Max for normalized numbers, may take longer when unnormalized.

n Number of shifts, $0 \leq n \leq 77$, (octal).

m Number of BCD characters (excluding sign) prior to terminate character $0 \leq m \leq 17$, (octal)

As an illustration of the use of the table we present a routine which packs the contents of A and R into A, the sign and first 26 bits of A retained, and followed by the last 13 bits of R. (This routine could be used for example in saving storage space for floating point numbers).

SYMBOLIC

LOC	OPR	ADD
PAK	SAX	N
	ADD	A BIT
	STA	EXIT .1
	XAR	
	ADD	EXCES
	STA	N
EXIT	CLA	N
	TRA	RTN

We might wish to use this with the following instructions (as calling sequence):

LOC	OPR	ADD
START	FCA	X
	TRA	PAK
END	STØ	X
	TRA	MØRE

Our hope in optimizing then becomes this: to arrange the routine "PAK" so that it can be completed in time to pick up the instruction pair at END without delay. For this we assume that the routine, "PAK", to be used, is in the V loop.

The optimum machine language coding for this then is found through use of Table I to be: (Assume we start at Loc 0004.0)

SYMBOLIC				MACHINE		
LOC	OPR	ADD	eff LOC	LOC	OPR	ADD
START	FCA	X	0004	0004	+30	00060
	TRA	PAK	0007		+57	77700
END	STØ	X	0005	0005	+60	00500
	TRA	MORE	0010		+57	00150
PAK	SAX	V	0014	7770	+15	77770
	ADD	A BIT	0017		+01	77710
	STA	EXIT .1	0025	7771	+42	77731
	XAR		0033		+43	00001
	ADD	EXCES	0042	7772	+01	77740
	STA	V	0050		+42	77770
EXIT	CLA	V	0063	7773	+00	77770
	TRA	RTN	0070		+57	00050
EXCES			const	7774	+00	00000
					-00	40000

Thus we see that the packing takes place within one drum revolution and hence is what we want.

d Difference in size of exponents $0 \leq d \leq 46$ (octal).

h Any address.

i Factor to be added (octal) to address of operand in left half command in order to determine effective address for right half instruction. If factor in this column has an * then it is to be added to address of instruction, not address of operand. (Same for absolute value commands.)

j Not optimizable.

k Factor to be added (octal) to address (or effective address) of instruction in order to select optimum address for operand.

m Number of BCD characters (excluding sign) prior to terminate character; $0 \leq m \leq 17$ (octal).

n Number of shifts; $0 \leq n \leq 77$ (octal).

* Factor to be added (octal) to left half instruction address rather than operand address to determine effective right half instruction address.

** These factors indicate maximum for normalized numbers, may take longer when unnormalized.

*** If the instruction under consideration is in a right half word then we cannot optimize the reading of the next instruction; however, the earliest it can be read (its effective address) is 02 greater than the effective address in the case of a left half word.

HOW TO USE TABLE I

Table I may be used both for optimizing and for computing expected running times of programs. It may be used both for programming outside of and within the high speed loops.

When using Table I for coding where loop addresses are involved, code just as in the case of main memory but replace addresses by their loop equivalents.

When the computer is in continuous mode of operation, the pressing of the Start button after the halt and transfer to 0003 has the following effect: The word at 0014 is picked up at sector time 0014, the transfer is made and instruction 0010.0(L) is picked up at sector time 24 and put into the C register. The store then takes place at sector time 0030 when the write head for Channel 77 is at 7770. Similarly, the store at 0010.1(L) takes place at Sector time 36. Thus, in continuous mode we are assured that both main memory locations 7770 and 7776 are changed. The program tests for this and when it is the case, types out the letter "C".

However, if the machine is now set to single step mode, the following sequence occurs beginning at 0003.1, (The word at 0014 is in the accumulator). The transfer is executed and the instruction at 0010 is placed in the C register; the machine halts. As soon as the start button is pressed the machine begins to execute the store into 7770. The probability is only one in eight that this store will occur at sector time 30, since it depends on the sector time when we release the start button. Similarly, the store at 0010.1(L) will probably not occur at sector time 36. Then we test to see if one of the above stores failed to occur and if so, we type out the letter "S". Thus, with a probability of 63/64 the machine can tell whether or not is is in single step mode. By adding further stores of this kind we can increase this probability to any level, (less than 1).

REFERENCES: Operating Manual for RECOMP II.

INFORMATION TO: All concerned

WRITTEN BY: Harry L. Nelson
Applied Mathematics

As a final example of the use of the tables we present the following routine, which, in its explanation, will show how the programmer may make use of the so-called gray-area; that is, main memory locations 7760 to 7777.*

Information may be entered directly into these locations by use of either tape, typewriter or console. Information contained therein may be observed by setting the dials to the desired location and pressing a readout button. However, information contained therein may not be typed or punched out**.

Internally, main memory locations 7760-7777 are addressable (by the reading heads) for the purpose of picking up information stored therein, only through the use of the commands CTL and CTV.*** However, for commands involving storage of information, the situation is different. The commands CFL and CFV always store information in main memory--never in the loops.**** The commands STØ, SAX, STA, and FST, when used with addresses 7760 to 7777, always store information in the loop area, and sometimes may also store information into main memory area 7760 to 7777. Moreover the instruction FST 7777 may store information into location 7700.

The circumstances which cause information being stored into the loops to be stored also into main memory will now be explained. When a command which specifies that information is to be written into memory is encountered, circuitry is set up to allow this to be done. For this purpose the section of main memory 7760-7777 is treated exactly the same as the last 16 words of any channel. However, when the proper instruction is given, in addition to preparing the circuits to write information into main memory locations 7760-7777 by use of the write head for channel 77, circuitry is also set up to write information into the corresponding loop area by use of the appropriate loop write head. As soon as the writing has been accomplished, these circuits are disengaged and the next command is performed. It may happen that the information is written into the loop at the same time that the main memory location bearing that address is under the main memory write-head. When this happens information is simultaneously recorded in both main memory and loop areas. By careful programming one can assure that this simultaneous writing will or will not happen. Thus the so-called gray area can be utilized by the programmer as needed.

* As we shall see location 7700 should also be included in the gray area.

** The commands PNW, TYW, PTW, address of 7757.1, use main memory locations 7757 & 7760 rather than loop 7760.

*** The commands FCA, FCS, FMP, FDV, FAD, FSB, and FSQ with the address of 7757, also use main memory 7760.

**** The commands CFL 776X and CFV 777X cause all words in the L and V loops, respectively, to be cleared to - zero; however, the former contents of L (or V) are transferred to main memory.

In, fact, this unusual property makes possible some strange effects. Programs may be written which will do the following: Allow the machine to decide whether the computer is in continuous or single step mode, allow the machine to decide whether or not the transfer stop switch is set, or to decide whether the preset stop switch is set, and if so, whether it is in position "1st" or "2nd". Programs whose output depends on the rapidity with which the start button is pressed.

LOCATION	COMMAND	ADDRESS	COMMENTS
0001	+64	00100	
	+00	77610	
0002	+57	00210	To get proper timing
	+00	00000	
0003	+00	00140	
	+57	77600	
0010	+60	77700	
	+60	77700	
0012	+03	77600	
	+52	77660	
0013	+00	77760	
	+03	77600	
0014	+52	77660	
	+72	00160	
0015	+57	00011	
	-00	00000	
0016	+72	00050	
	+57	00011	
0021	+60	77700	Thus contents of 0011 are put in main memory 7770 & 7776
	+60	77760	
0022	+77	00030	
	-00	00000	

HOW TO USE TABLE II

This table presents the data of Table I in a somewhat different form for the case in which the right half command is a transfer.

1. What to do if instruction pair is not in sector 00.

In this case add actual sector number to each address.

2. What to do if operand address not optimized.

Here, add difference between actual address and optimum (table) operand address to next instruction address.

Example: (Optimum program).

LOC	OPR	ADD	OPR	ADD	
0001 +	CLA	00030 +	TRA	01110	
0111 +	MPY	00130 +	TRA	35700	
3570 +	SUB	60720 +	TRA	01010	
0101 +	STA	77651 +	TRA	60220	
6022 +	STØ	00130 +	TRA	00600	*non-optimized since we wish to replace former multiplier. (optimum address is XX65).
0060 +	DIS	23620 +	TRA	01110	*non-optimized since we wish to loop. (Optimum address is XX31).

Other instructions not considered in Table II for which partial optimization is possible are TYW, PTW, and PNW. For each of these we can optimize the address of the operand, but not the address of the next instruction. This is due to the fact that here we have a mechanical function, which is not connected with the internal clocking of the computer. The correct optimization for the operand is the same as for DIS in each of these cases.

Another case in which optimization may be accomplished is in the use of trapping mode (negative commands). For negative commands in the left half of the command pair the optimum sector is 73. For right half negative commands the situation is the same as for TRA. Thus for example in location 0073, -CLA 0045 + TRA 0022 will trap to 0000.0 in 1.35 ms., while in 0074, -CLA 0076 + TRA 0004 takes 17.43 ms. to trap to zero.