MPRO-06

HAP REPORT #2

Subject: Harvest Assembly Program - HAP

By: Raymond W. Southworth

Date: September 30, 1959

HARVEST ASSEMBLY PROGRAM

Introduction

The set of formats for the Stream Byte-by-Byte instruction and its set-up words have been modified and expanded somewhat since HAP Report #1 was written, particularly to include some of the recent changes that have been made in the hardware of Harvest.

In addition the formats for the hybrid instructions are now included. An attempt has been made to keep them consistent with the format of the Stream Byte-by-Byte instruction. Also included in this report is a list of the system symbols and the mnemonics to be used in HAP.

The general ideas described in the previous report have been retained. For example, the block of 10 set-up words is considered in terms of functional units or fields which may be less than a machine word in length or which may even be in separate words. It has also seemed desirable from the programmer's viewpoint to write as part of the set-up some of the functions now represented in the Stream Byte-by-Byte instruction itself. Thus, the set-up word for the logic unit may include such related information as the operation code of the unit, the Single/Double code, and the Carry Propagate code. The assembly program will then insert these fields into the Stream Byte-by-Byte, as required in the machine word. Examples to illustrate this are given on page 7 of this report.

Although it would be helpful to the programmer to be able to write all his adjustments in one block, whether they follow immediately after the stream instruction or are part of a chain, no provision has been made for this as yet. Also, it is necessary for the programmer to be sure that adjustments in a chain are not placed beyond the allowable range of 255 half-words from the stream instruction.

1. Stream Byte-by-Byte

SBBB(connections), LUOP, group size, TA(mode, P/S, RBA), LU(S/D, CP), STOP(stimulus)

Ex: SBBB(P-LU, Q-LU, LU-R), EQT(P, NB), FL1P, STOP(FL2P)

Notes:

1. The connections may be given either in terms of the system symbols for the actual units connected or the numbers of the open gates.

2. If a field or sub-field is null, the corresponding bits will be set to zero. Thus, in the above example it is not necessary to write the S in the field for LU, since this corresponds to a zero bit.

3. The various fields may be written in any order, with the exception that SBBB(connections) must be the first field.

2. Adjustments (4 types)

ADJ#(stimulus), action 1, action 2, action 3

ADJ#(stimulus AND), action 1, action 2, action 3

ADJ#(stimulus), action 1, action 2, SKA, relative address

ADJ#(stimulus), action 1, action 2, CHAIN, relative address

Ex: ADJ1(NW.NY), REP(P,Q), RESET(SACC)

Notes:

1. The adjustments may be numbered if desired but they will be assembled in the order as written.

2. An action which has more than one operand, such as REP(P,Q) may be written either as above or separated as REP(P), REP(Q). It is the responsibility of the programmer to use no more than three actions in any one adjustment. Also, the actions will be assembled in the order written either explicitly or implied by the order of the grouped operands.

3. In the first type of adjustment, there are simply three action fields. In the second type, the stimulus of the next lower-order priority adjustment must also be present for the actions to take place. In the third type, there is a skip relative to the address of the stream instruction to an instruction in the arithmetic mode. In the fourth type there is a skip relative to the address of the stream instruction to another adjustment.

3. Set-up

SETUP

PART SETUP

Notes:

1. SETUP is a pseudo-op to be used whenever it is desired to write a full set-up of 10 words. It will reserve the block, beginning at a full word, set all words to zero, and then OR in the given fields. The following set-up words are self-identifying and may be written in any order. They are assumed to include all words up to the first word which is not a set-up word.

2. PART SETUP is used whenever it is desired to write less than the full 10 words. The following set-up codes will be examined, and a block of full words extending from the first non-zero word to the last non-zero word in assembled form will be reserved. Unused fields within this block will be set to zero.

4. Match Unit

WMAT (character, connections, R/F), IF(WOR), action

XMAT (character, connections, R/F), IF(XOR), action

YMAT (character, connections, R/F), IF(YOR), action

ZMAT (character, connections, R/F), IF(ZOR), action

Ex: XMAT((8)377, P.Q,F), IF(XOR), OMIT ALL

Notes:

1. The match character may be given in either octal or Hollerith by means of an entry mode.

2. The connections as used to cause a stimulus for adjustments, counters, and group size are indicated by a period between units when logical AND is meant and by a V for logical OR. The connections are assumed to be only of the OR type when considering the stimulus for the omitting (swallowing) of bytes, as indicated in the stimulus XOR.

3. The R/F bit indicates whether the match is on only the rightmost bit or on the full byte. A null field is assumed to mean a match on the full byte.

5. SCTR Unit

SCTR(limit, value), IF(stimulus), action

Ex: SCTR(1000), IF(XVY), SC + 1

Notes:

1. Limit and value must be written in the above order.

6. SACC Unit

SACC(threshold, value), IF(stimulus), action

7. Logic Unit

LU(modulus, S/D, CP), action

Ex: LU(150), CXXXX

Notes:

1. The modulus as assembled will be positioned so as to eliminate leading zeros, as required in the machine word.
2. If neither Double nor Single column operation is specified, S will be assumed.
3. If CP is not specified, there will be no carries propagated.
4. The possible actions are:

CXXXX	Connect, where XXXX is used as in STRAP
MAX(P,Q)	
MIN(P,Q)	
EQT(P,0)	Equals test
EQT(P,NB)	
EQT(0,P)	
EQT(NB,P)	
EQT(0,Q)	
EQT(NB,Q)	
GET(P-Q,0)	Greater than or equals test
GET(P-Q,NB)	
LET(Q-P,0)	Less than or equals test
LET(Q-P, NB)	
MOD(P-Q)	P-Q, mod M

(continuation of logic unit actions)

MOD(Q-P)	Q-P, mod M
BIN(P+Q)	P+Q, Binary (code 30)
MOD(P+Q)	P+Q, mod M (code 31)

5. In each of the test functions, the comparison is between the P and Q bytes, always in that order. If the test is satisfied, the output is the first byte given within parentheses; if it is not satisfied, the output is the second byte.

6. The connections to the logic unit may be P, SCTR, or TE; and Q. Thus, in all the above P may be replaced, if desired, by SCTR or TE if one of those is being used in place of P.

7. All of the fields except the modulus will be assembled into any stream instruction referring to this particular set-up.

8. F Unit

F(limit), IF(stimulus), action

Ex: F(2), IF(KK,LM),SO

Notes:

1. The stimulus in this case is written as the bit or bits to be tested. A one in any of them causes the counter to advance according to the indicated action.

9. Starting Addresses for Stream Units and Index Tables

PAD (stream address, index table address)

QAD (stream address, index table address)

RAD (stream address, index table address)

10. Table Assembly Unit

TA (mode, P/S RBA), TBA, TCS, TAP(TPS, TPI, TPN),TAQ(TQS,TQI,TQN)

Notes:

1. RBA indicates replacement of base address

2. TBA = table base address

3. TCS = cell size

4. TPS = initial offset of P stream

5. TPI = increment to be applied to offset for P stream

6. TPN = number of bytes from P stream

7. TQS, TQI, and TQN are similar to TPS, TPI, and TPN, but apply to the Q stream.

8. The mode and P/S and RBA fields will be assembled in the stream instruction.

11. Table Extract Unit

TE (mode, MDM), BM, TEI, TEN

Notes:

1. MDM if present indicates memory distributor mode

2. BM is the byte mask for TE

3. TEI is the increment for TE

4. TEN is the number of bytes to be extracted

12. Stream Stimulus Mask

SSM (mask)

13. Debug Code

DEBUG (4 bit code)

14. Index Words

```
P                                  M
QX (Mode, EC, FF, NR, RI, SS, R ), N,I,BM, offset, RBL
R                                  MR
                                   O
```

Notes:

1. If any of the one-bit codes, such as EC, is not specified, the opposite case, CC here, will be assumed. These codes need not be in any particular order.

2. For index words with flag bits, the codes PXF1, PXF2, and PXF3 should be used.

3. For a branch level word, the code PXBL should be used. The branch address is then wirtten in place of the byte mask.

4. The programmer fields N, I, etc., must be in the order shown. Null fields are indicated by a zero; field drop-out is from right to left.

The following examples are intended to illustrate the general procedure to be followed in writing a stream instruction.

Example 1:

Here the SBBB instruction is written with all the fields contained in the actual machine word.

```
     TI, 10, JOE, $HR                'Transmit set-up to registers
     BES, PETE
 JOE SETUP
     WMAT(character, connections), IF (WOR), action
     SCTR(limit, value), IF(stimulus), action
     SACC(threshold, value), IF(stimulus), action
     LU(modulus)
     F(limit), IF(stimulus), action
     TA, TBA, TCS, TAP(TPS, TPI, TPN)
     TE(mode, MDM), BM, TEI, TEN
     SSM(mask)
     DEBUG(code)
     PAD(stream address, index table address)
PETE SBBB(connections), LUOF, GS, TA(mode, P/S, RBA), LU(S/D, CP), STOP(stim)
     ADJ1(stimulus), actions
     ADJ2(stimulus), actions
     ADJ3(stimulus), actions
     ADJ4(stimulus), actions
     ADJ5(stimulus), actions
     B, address
     ADJ6(stimulus), actions
           etc.
```

Example 2:

Here the SBBB instruction is to be filled in from the set-up by the assembly program. Words in the set-up which are the same as those above have been omitted, but would, of course, have to be included in any real problem.

```
     TI, 10, JOE, $HR
     BES, PETE
 JOE SETUP
     LU(modulus, S), action
     TA(mode, P/S, RBA), TBA, TCS, TAP(TPS, TPI, TPN)
PETE SBBB(connections), GS, STOP(stimulus), SETUP JOE
     ADJ1(stimulus), actions
           etc.
```

Example 3:

Here the SBBB instruction is to be filled in partly from the "set-up".

```
     TI, 10, JOE, $HR
     BES, PETE
JOE  SETUP
     LU(modulus, S)
     TA(mode, P/S, RBA), TBA, TCS, TAP(TPS, TPI, TPN)
PETE SBBB(connections), LUOP, GS, STOP(stimulus), SETUP JOE
          etc.
```

Notes:

1. Overruling of the set-up as in Example 3 above may be done only for a complete field. Thus, LUOP, or TA(mode, P/S, RBA), or LU(S/D, CP) may be overruled, but not just the mode for TA, for example.

Formats for the Hybrid Instructions

SNOP, GS, STOP (stimulus)

SMER(D/A, UP/DN, I/E, S/O), STOP(stimulus)

SSER(D/A, ORD/RAN, UP/DN, S/O, SCD), STOP(stimulus)

Note:

The codes for SCD, the search condition, are

PLEQ	$P \leq Q$
PGEQ	$P \geq Q$
PLQ	$P < Q$
PGQ	$P > Q$
PEQ	$P = Q$
PNEQ	$P \neq Q$

SSEL(L/G, S/O), STOP(stimulus)

STIR(R/T, D/A, IC/DC, UP/DN, S/O), STOP(stimulus)

SMLU(PARG/RARG), GS, TA(mode, P/S, RBA), LU(S/D, CP), STOP(stimulus)

SQNL(P-T/Q-T), GS, TA(mode, P/S, RBA), LU(S/D, CP), STOP(stimulus)

SILS(L/S, CS), GS, TA(mode, P/S, RBA), LU(S/D, CP), STOP(stimulus)

Note:

CS is the cell size and is given by one of the following codes:

8B	8 bits
16B	16 bits
32B	32 bits
1W	1 word
2W	2 words
4W	4 words
8W	8 words
16W	16 words

HAP Mnemonics and System Symbols

Units

P	stream unit
Q	
R	
TA	table address assembler
TE	table extract unit
LU	logic unit
F	F unit
SCTR	statistical counter
SACC	statistical accumulator
WM	W match unit
XM	X
YM	Y
ZM	Z

Registers (all to be preceded by $, as in $HR)

$HR	harvest registers
WCHAR	W match character
WCON	W connections
WSPAN	W span bit
WOP	W operation
WMODE	OR/AND mode
XCHAR	
XCON	
XSPAN	
XOP	
XMODE	
YCHAR	
YCON	
YSPAN	
YOP	
YMODE	
ZCHAR	
ZCON	
ZSPAN	
ZOP	
ZMODE	
SSM	stream stimulus mask
SATH	SACC threshold
SASTEPSTIM	SACC stimulus
DEBUG	DEBUG code
SACC	
SCTR	

SAMODE	SACC mode
SCMODE	SCTR mode
SCSTEPSTIM	SCTR step stimulus
SCLIM	SCTR limit
F	F unit
TBA	table base address
CS	table cell size
MDM	memory distributor mode
TAPS	initial offset, P stream
TAPI	increment
TAPN	number of bytes
TAPJ	
TAPM	
TAPFS	
BST	bootstrap for TA
TAQS	initial offset, Q stream
TAQI	
TAQN	
TAQJ	
TAQFS	
MOD	modulus
TES	initial address, TE unit
TEI	
TEN	
TEM	
TEBM	byte mask
SSS	stream stimulus status
PS	start address, P stream
PIX	index table, P stream

QS	
QIX	
RS	
RIX	
ERRIND	error indicator register
BSP	boot strap for P
BSQ	boot strap for Q
BSR	boot strap for R
BSPC	boot strap, program controlled
TE	table extract unit
R1	R unit
R2	

Index Words

IXI	increment
IXIO	increment with offset
IXO	offset
IXNO	N/O bit
IXUL	U/L bit
IXTR	TR bit
IXRM	RM field
IXN	number of bytes
IXRBL	RBL bit
IXRN	number of bytes, using RBL
IXF1	flag 1
IXF2	flag 2
IXCC	CC bit
IXFF	FF bit
IXSR	SR bit

IXFS	FS bit
IXJ	J field
IXBM	byte mask
IXM	M field
IXBL	BL bit
IXNS	NS bit
IXBRHO	branch address, high order
IXBRLO	branch address, low order

Indicators in Sigma Interrupt Mechanism

Bit Address	Code	
11.15	OP	operation invalid
11.22	MCO	memory count overflow
11.23	EW	extract wraparound
11.28	SAOU	SACC overflow/underflow
11.29		
11.32		
11.33		
11.41	ECP	end chain P
11.42	ECQ	end chain Q
11.43	ECR	end chain R
	LST	lost stimulus
	SCOU	SCTR overflow/underflow
	AERR	arithmetic error

Stimulus Codes

NOP	
INIT	Initial
EL1P	End level 1, P
EL1Q	
EL1R	
EL2P	
EL2Q	
EL2R	
ELTE	
EG	End of group
ESQ	End of sequence
FL1P	Flag 1, P
FL1Q	
FL1R	
FL2P	
FL2Q	
FL2R	
FL3P	
FL3Q	
FL3R	
FFP	First-to-follow, P
FFQ	
FFR	
BLP	Branch level, P
BLQ	
BIR	

Stimulus codes (continued)

ECP	End chain, P
ECQ	
ECR	
ECTE	
BYP	Byte from P
BYQ	
BYR	Byte to R
OPLU	Operation in LU
+BYSA	+ byte into SACC
-BYSA	
SAGETH	SACC $\geq$ threshold
SABN	SACC becomes negative
SALTH.EG	SACC $<$ threshold, EG
SAGETH.BY	SACC $\geq$ threshold, byte into SACC
SCSTEP	SCTR steps
SCLIM	SCTR = limit
SCNLIM.EG	SCTR $\neq$ limit, EG
W	W match unit signal
X	
Y	
Z	
XVY	X or Y signal
W.X	W and X signal
W.Y	
NW	Not W
NX	
NY	
NZ	

Stimulus codes (continued)

NW.NY	Not W and not Y
NW.NX.NY.NZ	
F1	F = limit
F0	F ≠ limit
F0.EG	
KB0	KB = 0
KB1	KB = 1
LB0	
LB1	
MB0	
MB1	
KG0	
KG1	
LG0	
LG1	
MG0	
MG1	
KB0.F0	
KB1.F0	
KB0.F1	
KB1.F1	
LB0.F0	
LB1.F0	
LB0.F1	
LB1.F1	

Stimulus codes (continued)

MB0.F0	
MB1.F0	
MB0.F1	
MB1.F1	
KG0.EG	
KG1.EG	
LG0.EG	
LG1.EG	
MG0.EG	
MG1.EG	
MCO	Memory count overflow
SAOU	SACC overflow/underflow
LST	Lost stimulus
SCOU	SCTR overflow/underflow
EW	Extract wraparound
OP	OP invalid
AERR	Arithmetic error
ADJ1	Adjustment 1
ADJ2	
ADJ3	
ADJ4	
ADJ5	
ADJ	Any adjustment
ADJST	Adjustment stimulus
EC	End of chain
FL	Flag bit

Stimulus codes (concluded)

DEBUG	Debug signal
PERR	P error
QERR	
SERR	
TAERR	
TEERR	
LUERR	
MERR	Memory error
BYSA	Any byte into SACC
STOP	
UUA	Ungated unit adjusted

Adjustment Reactions

Note: Possible operands are shown in parentheses.

NOP	
DSA MU	Disable match units
DSA STIM THIS BY	Disable stimuli this byte
DSA STIM TO EG	Disable stimuli to end of group
RESET(P, Q, R, SACC, SCTR, TA, TE)	
RESET THRU FL1(P, Q, R)	
RESET THRU FL2(P, Q, R)	
RESET THRU FL3(P, Q, R)	
SKIP(R, TA, TE)	
SKIP THRU EG(LU)	
INSERT W IN L	
INSERT X IN L	
INSERT Z IN L	
INSERT MOD IN L	
INSERT MOD IN TE	
SC+1	Step SCTR by +1
SC-1	
SC+TBA	Add TBA to SCTR
SA+1	Step SACC by +1
ADV(P, Q, R)	Advance to next level
ADV NEXT ABOVE FL1(P, Q, R)	
ADV NEXT ABOVE FL2(P, Q, R)	
ADV NEXT ABOVE FL3(P, Q, R)	
REP(P, Q, TE)	Repeat byte

Adjustment reactions (concluded)

RUN TO R(P, Q, TE)	Run out
RUN TO R THRU FL1(P, Q)	
RUN TO R THRU FL2(P, Q)	
RUN TO R THRU EG(TE)	
RUN TO SACC(TE)	
OMIT THIS BY(LU, R)	Swallowing action
OMIT NEXT BY(P, Q, TE)	
MATCH (P, Q, TE)	
MATCH THRU FL1(P, Q)	
MATCH THRU FL1(P, Q)	
MATCH THRU FL2(P, Q)	
MATCH THRU EG(TE)	
RO8(SACC, SCTR)	Read out
RO16(SACC, SCTR)	
RO24(SACC)	
RO8R(SCTR)	Read out and reset
RO16R(SCTR)	
CANCEL (TA)	
PUT TBA-1 FOR TAD	Substitute TBA-1 for T address
REF TBA-1 AND INC	Reference TBA-1 and increment