MPS/CO Computer 71-8
28 September, 1971

CERN MPS PDS1 ASSEMBLY LANGUAGE PDSA

B. Carpenter

Introduction

The language and assembler program described here were developed for the Imlac PDS1 display mini-computers attached on-line to the CERN PS IBM 1800. Programs are written in a language based on the Imlac PDS1 assembler, and are punched on cards in a similar format to the 1800 Assembler. They are assembled in the 1800 as background-processing jobs under MPX. The assembler program (PDSA) makes two passes through the program, giving a listing on the 1443 lineprinter during Pass 2 if requested. The object code is stored on the IBM disc during Pass 2.

If no errors occur, the assembled program may be built into a PDS1 Core Load (PCL) and stored in the PDS1 file on the IBM disc. However, space in this file is limited, so PCL's should not be built except when necessary.

This document should be read in conjunction with the Imlac manuals, but where there are differences this version is correct.

Format of programs

A program consists of a deck of statement cards and assembler control cards. At the end there must be an END statement, followed optionally by *BUILD control cards, followed by a *CCEND control card.

/je

The program is assembled by putting the following sequence of cards in the reader, and pressing CONSOLE INTERRUPT with sense switch 7 up:

```
// JOB   000001111122222
// XEQ   PDSA FX
(Program deck)
// END OF ALL JOBS
Blank.
```

Do not mix PDSA assemblies in the same job as //ASM assemblies; the two assemblers use the same part of the disc for working storage.

An example program is given at the end of this document.

Statement cards

These are of two types:

1) comment card. This has * in Column 21 and is ignored

2) operation card. This is in the following format:

Col. 21-25	Col. 27-30	Col. 32	Col. 35-71
LABEL	OPER	A	EXPR COMMENT

LABEL is up to 5 characters of which the first is a letter, and none of which is +, &, - or a space. When a new label is encountered by the assembler during Pass 1, it is added to the Temporary Symbol Table and assigned the value of the Current Location Counter (CLC), i.e. the absolute core location in the PDS1 of the instruction (except in EQU statements, see below).

OPER is 3 or 4 letters which must be a recognised operation mnemonic, i.e. either a pseudo-operation (see below) or a PDS1 instruction code stored in the Permanent Symbol Table. (See Appendix for a list of PDS1 instructions.)

A is either a space (normal) or I (in indirectly addressed instructions). Note that the auto-index registers, octal addresses)10 to)17 in each 2 K, increment by 1 each time they are used as an indirect address. The increment occurs before the operand is accessed.

EXPR consists of a string of elements, linked by + (&) and - signs and terminated by a space (except in DEIM and INC statements, see below).

Allowed elements are:

1) decimal numbers $\leqslant 32767$

2) octal numbers, indicated by a) before them, $\leqslant)177777$. (Note that the right bracket may be punched as 12, 8, 4 or as 11, 8, 5)

3) symbols. Any entry from either the Permanent or the Temporary Symbol Table, i.e. any PDS1 instruction mnemonic or any user-defined label

4) . (full stop), representing the value of CLC, the address in core of this instruction.

Thus valid expressions are:

```
LABEL
LABEL+3
.-1
.-.
LABEL+)17
)772
)76-62
CLL.
```

Assembly of PDS1 instructions

The object code created is the binary equivalent of OPER, OR ed with the indirect bit if requested, OR ed with some or all bits of the binary value of EXPR. For addressed instructions, EXPR must be a valid address of which the bottom 11 bits are used for mini-computer instructions and the bottom 12 bits are used for display processor instructions. For other instructions EXPR must have a suitable octal value (see Appendix).

Micro-instructions may be combined, where meaningful, by writing the first in the OPER field and the second in the EXPR field, but octal values must be used to combine more than two (see Appendix).

The value of CLC is increased by one after the assembly of each PDS1 instruction.

Pseudo-operations

OPER can be a pseudo-op. Some pseudo-ops change the value of CLC and some do not.

ORG EXPR

This statement changes CLC to the value EXPR. EXPR must be a valid core address, e.g.)2000.

There must be an ORG statement at the beginning of the program. There may be other ORG statements (but see the note in the description of the BSS statement).

BSS EXPR

This statement reserves a number of storage locations equal to the value of EXPR, and increases CLC by the same amount. It may be labelled. There is not BES statement. Note that ORG

statements and long BSS areas (more than 20 locations) cause a "file break" to be inserted in the object code. This corresponds to the start of a separate block transfer to the PDS1 when the program is eventually loaded for execution. A file break is also inserted automatically after approximately 240 words of object code have been generated since the previous file break. The total number of file breaks allowed is limited. The letters "FB" are printed at the extreme right of the listing of each line which causes a file break.

Short BSS areas (less than or equal to 20 locations) are set to contain zeroes.

DC EXPR

This location is set to contain the value of EXPR. This statement may be labelled. CLC is increased by 1.

EBC .CHARACTERS.

Packed EBCDIC characters (two per word) are inserted in the object code. The last half-word is filled with a space if necessary. Do not use a full stop in the COMMENT. This statement may be labelled. CLC is increased by the number of words generated.

Note the following punching requirements for correct recognition of characters in the PDS1:

+	12, 8, 6	or 12	(the second gives & on listing)
:	8, 2	(not printed on listing)	
(	12, 8, 5	NOT 0, 8, 4	as on computer room punches
)	11, 8, 5	NOT 12, 8, 4	as on computer room punches
=	8, 6	NOT 8, 3	as on computer room punches
?	0, 8, 7		
>	0, 8, 6	(not printed on listing)	
<	12, 8, 4		
%	0, 8, 4		

"new line" represented by $ or by 11, 9, 5

"tab" represented by ' or by 12, 9, 5

(11, 9, 5 and 12, 9, 5 will give correct 1053 code if converted by EBPRT, but are not printed on listing).

LABEL EQU EXPR

LABEL is added to the Temporary Symbol Table with the value EXPR.

DEIM BYTE

Equivalent to INC E/BYTE.

INC BYTE1/BYTE2

The assembler generates a word of object code consisting of two display processor increment mode bytes. CLC is increased by 1. The statement may be labelled. The bytes are coded in the EXPR field, separated by / and terminated by a space. Zero is assembled for missing bytes.

Byte codes:

Single letter : E (first byte only). Enter increment mode

N Exit increment mode, X and Y LSB zero

R Exit increment mode, X and Y LSB zero, display return jump

F Exit increment mode, X and Y LSB zero, X MSB + 1, display return jump

P Pause 2 μsec.

Multiple : Format L+n+m

L = B Bright vector

L = D Dark vector

+ (&) or - Sign of ΔX, ΔY

n = 0, 1, 2, 3 ΔX

m = 0, 1, 2, 3 ΔY

```
Thus        INC     E/B+3+3
            INC     B+3-3/F
```

draws an inverted "v" shape and positions the beam for the next character.

END

The last statement in a program.

Assembler control cards

These cards have a * in Column 1. The following are recognised:

*LIST List from here on
*UNLIST Stop listing
} may be used as often as required

*DELETE SYMBOL TABLES

Deletes all symbols (instruction mnemonics and labels). Does not affect permanent symbol table on disc.

*DEFINE PERMANENT SYMBOLS

All symbols defined so far (mnemonics previously in the permanent symbol table and labels) become "permanent" for the rest of this assembly. Does not affect permanent symbol table or disc.

*BUILD -XXXXX +YYYYY F EXPR

These cards follow the END statement. They are ignored if there were any errors.

Deletion of a PCL is achieved by placing a minus sign in Column 21 and following it immediately with the name XXXXX of the PCL to be deleted.

Building of a PCL is achieved by placing a plus (or &) sign in Column 27 and following it immediately with the name YYYYY of the PCL to be built.

F (Column 32) is normally a blank, but it is C for a PDS1 cold-start program and R for a PCL which is to be loaded into the PDS1 but not entered. (Instead the PDS1 will return to its previous mainline program, setting a skeleton flag.)

EXPR (Column 35) is the entry point in the PDS1. There must be a non-zero entry point punched even if Column 33 contains R or C.

*CCEND

Last card in the source deck.

Errors

The following errors cause immediate abortion with an message:

No END card
No *CCEND card (detected by MPX)
Input/output check
Symbol table full (too many labels)
Too many file breaks ("PROGRAM TOO LONG").

Errors during *BUILD operations also cause a message print-out and abortion.

After PDSA has aborted, the remaining cards of the source deck may cause some MPX error messages to be printed.

Other errors cause a two-letter error code to be printed on the listing at the extreme right, next to the "FB" printed for a file break.

The two-letter error codes are:

IC Illegal Character. A character has been found in a position where it is not allowed. (Note that unused fields are not checked, e.g. Col. 32 is ignored completely in DC statements.)

ID Illegal Definition. Label already defined. (Note that instruction mnemonics, i.e. permanent symbols, may not be used as labels.)

IO Illegal Operand in ORG or BSS statement

US Undefined Symbol

PE Page Error. Addressing error or operand error, especially attempting to address the wrong 2 K page

EO Expression Overflow. Value of expression could not be computed in 16 bits

TL Too long. Expression or element of expression contains too many characters.

Note: Symbols defined more than once are indicated by ID, but references to the symbols are not indicated as errors. The first definition is treated as valid. Where one card produces more than one error, only the last of them to be detected is shown on the listing.

B. Carpenter

Distribution

S. Battisti
H. van der Beken
G. Benincasa
L. Bertuzzi
J. Bosser
M. Bouthéon
L. Burnod
G. Cuisinier
A. Daneels
H. von Eicken, DD
P. Heymans
D. Lamotte
D. Lord, DD
J.H.B. Madsen
K.O.H. Pedersen
C. Piney, DD
C. Poinard
J.P. Potier
E. Ratcliff
W. Remmer
K. Schindl
C. Serre
G. Shering
U. Tallgren
D. Williams, DD

APPENDIX

INSTRUCTION MNEMONICS AND BRIEF DESCRIPTIONS

Processor Order

Operation field	Operand field	Function
LAW	EXPR	Load value of EXPR into AC. Must be positive, not more than 11 bits (2047)
LWC	EXPR	Load minus value of EXPR into AC. Value must be positive, not more than 11 bits (2047)
JMP	ADDR	Load value of ADDR into PC (Jump to ADDR). Note. In this and all addressed processor orders, ADDR must be in the same 2 K page as the instruction (i.e. bits 0 → 4 of ADDR and PC must agree). Indirect addressing is necessary for locations not in the same 2 K page.
DAC	ADDR	Contents of AC to store location ADDR (does not change accumulator)
XAM	ADDR	Exchange contents of AC and ADDR
ISZ	ADDR	Increment contents of ADDR by 1. Skip next instruction and complement link if contents become zero
JMS	ADDR	Store address of next instruction in ADDR. Jump to ADDR+1. (Subroutine entry)
AND	ADDR	Logical AND contents of AC and ADDR. Result in AC
IOR	ADDR	Logical INCLUSIVE OR contents of AC and ADDR. Result in AC
XOR	ADDR	Logical EXCLUSIVE OR contents of AC and ADDR. Result in AC
LAC	ADDR	Load AC with contents of ADDR
ADD	ADDR	Add Contents of ADDR to contents of AC — Result in AC, in 2's complement. Carry, if any, complements Link bit
SUB	ADDR	Subtract contents of ADDR from contents of AC — Result in AC, in 2's complement. Carry, if any, complements Link bit
SAM	ADDR	Skip if contents of AC equal contents of ADDR

Class 1 Micro-instructions

Note. For valid combinations, see Imlac manuals. PDSA does not, however, recognise the combination of HLT with other instructions.

HLT		Halt
OPR		No operation (2 μsec)
CLA		Clear AC
CMA		1's complement AC
STA		Set AC to -1, i.e.)177777
1AC		Add 1 to AC
COA		Set AC to +1
CIA		2's complement AC (AC = -AC)
CLL		Clear Link bit
CML		Complement Link bit
STL		Set Link bit
ODA		IOR AC and DS (Data switches)
LDA		Load AC with DS
CAL		CLA CLL

Class 2 Micro-instructions

RAL	N	Rotate AC and Link left N bits (N = 0, 1, 2, 3)
RAR	N	Rotate AC and Link right N bits (N = 0, 1, 2, 3)
SAL	N	Shift AC left N bits. Shift in zeros, do not change sign bit (bit Ø). (N = 0, 1, 2, 3)
SAR	N	Shift AC right N bits. Propagate sign bit at left (N = 0, 1, 2, 3)
DON		Turn display processor on

Class 3 Micro-instructions - Skips

Note. For valid combinations, see Imlac manuals

		SKIP IF:
ASZ		AC is zero
ASN		AC is non-zero
ASP		AC is positive
ASM		AC is negative
LSZ		Link bit is zero
LSN		Link bit is non-zero
DSF		Display is on
DSN		Display is off
KSF		Keyboard flag is set
KSN		Keyboard flag is not set
RSF		TTY has input data
RSN		TTY has no input
TSF		TTY has finished transmitting
TSN		TTY has not finished
SSF		33 1/3 Hz sync. on
SSN		33 1/3 Hz sync. off
HSF		Paper tape reader has data
HSN		PTR has no data

Input/Output instructions

IOT	)ABP	Any I/O order
DLZ		Display Program Counter set to zero
DLA		Contents of AC to DPC
CTB		Clear TTY Break
DOF		Turn Display Processor off
KRB		Keyboard read character
KCF		Keyboard clear flag
KRC		KRB KCF
RRB		TTY read character
RCF		TTY clear flag (input)
RRC		RRB RCF
TPR		TTY print character
TCF		TTY clear flag (output)
TPC		TPR TCF
HRB		PTR read character
HOF		Stop PTR
HON		Start PTR
STB		Set TTY Break
SCF		Clear 33 1/3 sync. flag
IOS		IOT Sync. (maintenance)
IOF		Disable Interrupt
ION		Enable Interrupt
PPC		Punch AC and clear flag
PSF		Skip if punch ready

Display Processor Orders

DLXA	EXPR	Load X register with EXPR	12 bits
DLYA	EXPR	Load Y register with EXPR	
DJMS	ADDR	Store address of next display processor instruction in DT register. Jump to ADDR	
DJMP	ADDR	Jump to ADDR Note. ADDR is 12 bits for DJMS and DJMP, so 4 K may be addressed directly. No indirect addressing	

Display processor micro-instructions

DHLT		Halt display processor
DOPR		Display no-op (2 μsec)
DSTS	N	Set scale for vector drawing N = Ø - scale ½ N = 1 - scale 1 N = 2 - scale 2 N = 3 - scale 3
DSTB DSTB	Ø 1	Set block Ø or 1 (8 K machine only)
DRJM		Display return jump (jump to address stored in DT register by DJMS)
DIXM		Increment X register Most Significant Bits by 1
DIYM		Increment Y register M.S.B.
DDXM		Decrement X register M.S.B.
DDYM		Decrement Y register M.S.B.
DHVC		High Voltage Sync. & continue
DDSP		2 μsec Intensify (for point plotting)
DHVH		High Voltage Sync. & Halt

PDSA LISTING
*LIST

```
                    *SIMPLE EXEC TEST
                    *
                    *
                    *
003130    000000            ORG     FSTAD                                  FB
003130    035417    GO      JMS     DELLN
003131    000002            DC      2
003132    000062            DC      50
003133    035330            JMS     INSST
003134    000002            DC      2
003135    000001            DC      1
003136    003224            DC      WAREA
003137    036044            JMS     MONIT
003140    100001            CLA
003141    035625            JMS     ERMON
003142    003242            DC      EAREA
003143    036044            JMS     MONIT
003144    036414            JMS     DRGRA
003145    003250            DC      AXAR
003146    036044            JMS     MONIT
003147    036414            JMS     DRGRA
003150    003265            DC      PTAR
003151    036044            JMS     MONIT
                    *
003152    007202            LAW     LT1
003153    022000            DAC     UIESC
003154    104012            LWC     10
003155    023223            DAC     CNT
003156    001132            IOT     )132      CLEAR
003157    004003            LAW     3
003160    023221            DAC     GOOF
003161    001141            IOT     )141
003162    100001            CLA
003163    023222            DAC     FLAG
003164    100001    LAB     CLA
003165    027222            XAM     FLAG
003166    102001            ASN
003167    013164            JMP     LAB
003170    100001            CLA
003171    035625            JMS     ERMON
003172    003322            DC      MESS
003173    033325            ISZ     MESS&3
003174    033223            ISZ     CNT
003175    013212            JMP     LAB1
003176    001161            IOF
003177    004002            LAW     2
003200    001141            IOT     )141
                    *
003201    012007            JMP     EXIT
                    *
003202    100001    LT1     CLA
003203    001132            IOT     )132
003204    100001            CLA
003205    027221            XAM     GOOF
003206    102001            ASN
003207    010173            JMP     BACK
003210    023222            DAC     FLAG
003211    010173            JMP     BACK
003212    100010    LAB1    CLL
003213    024022            XAM     SKF&2     WAIT FOR CHARACTER
003214    102001            ASN
003215    013213            JMP     .-2
003216    004003            LAW     3
003217    023221            DAC     GOOF
```

```
PDSA LISTING
003220    013164          JMP     LAB
003221    000000    GOOF  DC      .-.
003222    000000    FLAG  DC      .-.
003223    000000    CNT   DC      .-.
                    *
                    *
003224    000015    WAREA DC      EW-WAREA-1
003225    141100          EBC     .TEST MESSAGE$NEW LINE'TAB.
003242    000000    EW    BSS
003242    000005    EAREA DC      EE-EAREA-1
003243    153331          EBC     .TEST ERROR.
003250    000000    EE    BSS
003250    000000    AXAR  DC      0
003251    000001          DC      1
003252    000055          DC      45
003253    000001          DC      1
003254    000000          DC      0
003255    001750          DC      1000
003256    000120          DC      80
003257    000012          DC      10
003260    177470          DC      -200
003261    000310          DC      200
003262    000031          DC      25
003263    000050          DC      40
003264    000000          DC      0
                    *
003265    000000    PTAR  DC      0
003266    000000          DC      0
003267    000002          DC      2
003270    000000          DC      0
003271    000001          DC      1
003272    000005          DC      5
003273    000012          DC      10        XY LIST 1
003274    177766          DC      -10
003275    000024          DC      20
003276    177766          DC      -10
003277    000144          DC      100
003300    000014          DC      12
003301    000310          DC      200
003302    000310          DC      200
003303    001440          DC      800
003304    000000          DC      0
003305    000001          DC      1
003306    177777          DC      -1
003307    000005          DC      5
003310    000012          DC      10        XY LIST 2
003311    000012          DC      10
003312    000024          DC      20
003313    000012          DC      10
003314    000144          DC      100
003315    177764          DC      -12
003316    000310          DC      200
003317    177470          DC      -200
003320    001452          DC      810
003321    000000          DC      0
                    *
*LIST
003322    000003    MESS  DC      3
003323    040360          EBC     .HIT  0.
                    *
000000    000000          END
BUILT OK
END PDSA
```