A scientific computer — 4

More programming in high and low level languages

by J. H. Adams, M.Sc.

THE MORTGAGE PROGRAM in Table 8 computes, from a given principal, annual interest rate and period for which a loan is to run (represented by P, I and T in the program), the monthly repayment and repayment schedule for a standard mortgage. The format closely follows that of standard BASIC. In line 6, an interest factor

$$K=1+\frac{I}{100}$$

is calculated, whilst the expression evaluated in line 7 is

$$B=\frac{K^T}{K^T-1}\times\frac{IP}{1200}$$

using the stack operation ENT to push K^T into the Y and Z registers of the stack as shown in Table 9. A special print format is used in lines 13 and 19 to round the displayed values of B and P to the nearest penny.

Table 9 Stack operations for the mortgage program.

COMMAND	X	Y	Z	T
YX	K^T	-	-	-
ENT	K^T	K^T	-	-
ENT	K^T	K^T	K^T	-
1	1	K^T	K^T	-
-	K^T-1	K^T	-	0
/	$\frac{K^T}{K^T-1}$	-	0	0

Table 10 shows two separate programs cascaded into the programming area. The first is run by the command RUN 4 and is a game which simulates the landing of a rocket on Earth. Lines 4 to 8 set a fuel level of 120 (F), a velocity of −50m/s (V) and an initial height of 250m(H). After presenting this information, the computer waits for the player to type in a one second burn of fuel, B, which is checked against the present amount of fuel (line 14) and then used to reduce the velocity by B−5, provided that there is enough fuel available.

The aim, of course, is to simultaneously reduce the velocity and height to zero, without running out of fuel. The

Table 8 Print out of a mortgage program based on the high level language.

```
003 PRINT "                    •••MORTGAGE PROGRAM•••"
004 PRINT "    INPUT PRINCIPAL, INTEREST RATE & TERM"
005 INPUT P I T
006 LET K=I 100 / 1 +
007 LET B=K T YX ENT ENT 1 - / I • 1200 / P •
013 PRINT "  MONTHLY REPAYMENT ="B2
014 LET B=B 12 •
015 PRINT "       •••REPAYMENT SCHEDULE•••"
016 FOR X=1 STEP 1 UNTIL T
018 LET P=P K • B -
019 PRINT "AFTER"X0, " YEARS YOU OWE"P2, " POU..."
021 NEXT X
024 END

 ODED
```

Table 10 Cascaded programs for a rocket landing game and the solutions of F (x)=0.

```
004 LET C=0
005 LET F=120
006 LET V=-50
007 LET H=250
008 PRINT "HEIGHT="H2, "M  VELOCITY="V2, "M/S  FUEL LEFT="F0, "UNITS"
009 LET C=C 1 +
010 INPUT B
011 IF C<15 THEN 14
012 ERASE
013 LET C=0
014 IF B>F THEN 25
015 LET F=F B -
016 LET B=B 5 -
017 LET J=H
018 LET H=H V + B 2 / +
019 IF H<0 THEN 30
020 LET V=V B +
021 IF H=0 THEN 37
024 GOTO 8
025 PRINT "OUT OF FUEL  PREPARE TO CRASH."
026 END
030 LET V=V SQ J 10 • + RT
031 PRINT "YOU HAVE CRASHED AT"V2, "M/S."
032 END
035 PRINT "WELL DONE, YOU HAVE LANDED."
036 END
037 IF V=0 THEN 35
038 PRINT "YOU HAVE LANDED TOO FAST.  HAVE A NICE STAY "
039 END
098 PRINT "  THIS PROGRAM USES NEWTONS METHOD FOR SOLVING"
099 PRINT "THE FORMULA F = F(X).  ENTER AN INITIAL GUESS FOR X NOW. . ."
100 INPUT Q
101 ERASE
102 LET X=Q
105 GOSUB 200
110 LET G=F
115 LET X=X 1.00001 •
120 GOSUB 200
125 LET T=G SQ RT
127 TOP
130 IF T<0.000001 THEN 190
135 LET Q=1 F G / 1 - REC 0.00001 • - Q •
137 PRINT Q8
140 GOTO 102
190 PRINT "THE SOLUTION IS X="Q6
195 END
200 LET F=X LN X 3 • + 10.8074 -
205 RETURN

100F
```

exercise is based upon standard Newtonian equations of motion; $s=u+\frac{1}{2}a$ and $v=u+a$. Crash velocities are worked out (line 30), using $v^2=u^2+2as$. In the program execution, C acts as a go counter, clearing the screen every 15 burns. This might seem unnecessary, as it takes some unusual playing to avoid a crash and not win in that number of attempts. There is a simple technique for predicting solutions to this game, but I will leave the reader to deduce this.

One of the most economical solutions uses burns of 0, 0, 0, 25 and 50. For a more daunting version, the 2 in line 18 can be made an inputted variable (which will affect the acceleration due to gravity) or, even more difficult, a function of the value of H.

The second program uses Newton's method to solve the equation $F(x)=0$. The equation in this case, $\mathrm{Ln}(X)+3X-10.8074=0$, is written at line 200 and, as it is required twice in the program, it is called as a subroutine at lines 105 and 120. Given an initial guess Q, at line 100, the computer calculates the next guess at Q by

$$Q-\frac{F(Q)}{F'(Q)}$$

calculated by the approximation

$$Q1-\frac{0.00001F(Q)}{F(1.00001Q)-F(Q)}$$

Line 125 assigns the absolute value of G to T, G being the difference between two successive values for Q and, if T is below the criterion of accuracy set in line 130, the program branches to line 190 and prints out a final rounded solution for X.

Note that if these two programs, or any material with more than 31 lines, are loaded, a LIST or DEL command will list the first 31 and then display LIST INCOMPLETE, preceded by the next valid line number on the top line of the screen. To display the rest of the program, or the next 31 lines, press the space bar.

Scientific numbers

The computer switches to a scientific display on numbers greater than 99,999,999 or less than 0.0001. Numbers appearing in programs or being entered in response to an INPUT line, may be entered scientifically or in floating point, provided that they are within the computers range. When entering scientifically expressed numbers, a space is not required at the end of the figures because the E entered in the figures tells the computer that only two more digits are to be entered. The standard form of one figure in front of the decimal point will always occur in displayed results, but need not be adhered to when entering because the computer recognises 1.00E02, 100E00, 0.01E04, .001E05 or 1000000E-04 as all being 100. This is demonstrated in the next program. Fig. 19 shows a recommended circuit for the Motorola MC1303 dual amplifier used as a RIAA equalised phono pre-amplifier. Tables 11 and 12 show the program for, and a run of, an analysis of the circuit. Values are entered in the most convenient units, resistors in kilohms, D and E in picofarads, and F in microfarads, and then scaled to their basic units in lines 8 to 23. The equations for working out the gain at various frequencies are;

$$G=1+(WDA)^2$$

$$H=1+(WEB)^2$$

$$I=\frac{A^2D}{G}+\frac{B^2E}{H}$$

$$J=\frac{A}{G}+\frac{B}{H}$$

$$K=WCF$$

$$L=\frac{((J-WKI)^2+(JK+WI)^2)^{\frac{1}{2}}}{C(K+1/K)}$$

The last equation is a good argument for Reverse Polish. Note that in; line 26 π can be called as PI.

Table 11 Program for analysing the pre-amplifier in Fig. 19.

```
005 INPUT A D B E C F
006 PRINT "FREQ            GAIN (DB)"
008 LET A=A 1000 •
011 LET B=B 1000 •
014 LET C=C 1000 •
017 LET D=D 1E12 /
020 LET E=E 1E12 /
023 LET F=F 1E06 /
026 LET W=PI 2 •
028 LET G=W D A • • SQ 1 +
030 LET H=W E B • • SQ 1 +
032 LET I=A SQ D • G / B SQ E • H / +
034 LET J=A G / B H / +
036 LET K=W C F • •
038 LET L=J W K I • • - SQ W I • K J • + SQ + RT C / K K REC + /
040 LET L=L 60 / LOG 20 •
044 LET M=W 2 / PI /
047 IF M>20000 THEN 200
048 PRINT M1 L0
052 LET W=W 10 RT RT
053 GO 28
200 END

0E15 ,
```

Table 12 Computer run of results for the pre-amplifier.

FREQ	GAIN (DB)
1.0	6.
1.8	11.
3.2	15.
5.6	19.
10.	21.
17.8	21.
31.6	19.
56.2	16.
100.	12.
177.8	8.
316.2	4.
562.3	2.
1000.	0.
1778.3	-2.
3162.3	-5.
5623.4	-9.
10000.	-13.
17782.8	-18.

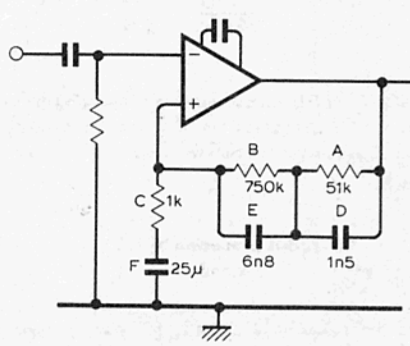

Fig. 19. *Typical RIAA equalised pre-amplifier based on the MC1303. The results of a computer run on this circuit are shown in Table 12.*

Table 13 Program for computing the intercept and gradient of the best fitting straight line.

```
001 LET A=0
002 LET B=0
003 LET C=0
004 LET D=0
005 LET N=0
006 LET E=0
007 INPUT X Y
008 LET N=N 1 +
009 LET A=A X +
011 LET B=B Y +
013 LET C=C X Y • +
015 LET D=D X SQ +
016 LET E=E Y SQ +
018 LET M=C A B • N / - D A SQ N / - /
019 LET L=B N / M A • N / -
021 PRINT 'AFTER'NO, 'PAIRS, M='M7 'C='L7
023 LET R=C A B • N / - SQ D A SQ N / - / E B SQ N / - /
024 LET R=R 100 •
025 PRINT 'COEFFICIENT OF DETERMINATION='R2, '%  '
027 TOP
029 GO 7

0DA7
```

line 38 full use is made of the 4 level stack for storing intermediate values and results. This line actually consists of more than one v.d.u. line's worth of characters, and it therefore overruns into the next line.

line 38 & 52 RT is an abbreviation of ROOT. In word recognition, the computer only considers the first and last letters of a word, which allows for considerable laxity in typing.

When establishing the relationship between two sets of data, the first test is usually one of proportionality, i.e. will the data, if plotted, give a straight line? Table 13 lists a program which uses linear regression to compute the intercept and gradient of the best fitting straight line for a series of pairs of co-ordinates (horizontal, then vertical), read in at line 7. Each set of data updates the values of M and C, and also takes part in the calculation of a coefficient-of-determination, which gives a measure of the fit of the line to the co-ordinates. Note the use of the command TOP at line 27, which clears and resets the data entry point to the top of the screen each time.

Low level programming

When low level programming is used, charts of the type shown in Table 14 are very helpful for translating between the mnemonics for the Z80 operations and the actual hexadecimal codes. If the charts are used in conjunction with the technical manual for the MK3880/Z80, program assembly and disassembly is

Table 14 Conversion charts for the Z80 instruction set.

Second character of Z80 code

First character	0	1	2	3	4	5	6	7
0	NOP	LD BC,nn	LD(BC),A	INC BC	INC B	DEC B	LD B,n	RLC A
1	DJNZ	LD DE,nn	LD(BC),A	INC DE	INC D	DEC D	LD D,n	RL A
2	JRNZ,e	LD HL,nn	LD(nn),HL	INC HL	INC H	DEC H	LD H,n	DAA
3	JRNC,e	LD SP,nn	LD(nn),A	INC SP	INC(HL)	DEC(HL)	LD(HL),n	SCF
4	LD B,B	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A
5	LD D,B	LD D,C	LD D,D	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A
6	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A
7	LD(HL),B	LD(HL),C	LD(HL),D	LD(HL),E	LD(HL),H	LD(HL),L	HALT	LD(HL),A
8	ADD B	ADDC	ADD D	ADD E	ADD H	ADD L	ADD(HL)	ADD A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB(HL)	SUB A
A	AND B	AND C	AND D	AND E	AND H	AND L	AND(HL)	AND A
B	OR B	OR C	OR D	OR E	OR H	OR L	OR(HL)	OR A
C	RET NZ	POP BC	JPNZ,nn	JP,nn	CNZ,nn	PUSH BC	ADD n	RST 0
D	RET NC	POP DE	JPNC,nn	OUT A,(N)	CNC,nn	PUSH DE	SUB n	RST 16
E	RET PO	POP HL	JPPO,nn	EX(SP),HL	CPO,nn	PUSH HL	AND n	RST 32
F	RET P	POP AF	JPP,nn	DI	CP,nn	PUSH AF	OR n	RST 48

First character	8	9	A	B	C	D	E	F
0	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	RRC A
1	JR,e	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	RR A
2	JRZ,e	ADD HL, HL	LD HL,(nn)	DEC HL	INC L	DEC L	LD L,n	CPL
3	JRC,e	ADD HL,SP	LD A,(nn)	DEC SP	INC A	DEC A	LD A,n	CCF
4	LD C,B	LD C,C	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
5	LD E,B	LD E,C	LD E,D	LD E,E	LD E,H	LD E,L	LD E,(HL)	LD E,A
6	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
7	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
8	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC(HL)	ADC A
9	SBC B	SBC C	SBC D	SBC E	SBC H	SBC L	SBC(HL)	SBC A
A	XOR B	XOR C	XOR D	XOR E	XOR H	XOR L	XOR(HL)	XOR A
B	CP B	CP C	CP D	CP E	CP H	CP L	CP(HL)	CP A
C	RET Z	RET	JPZ,nn	*	CZ,nn	CALL,nn	ADC n	RST 8
D	RET C	EXX	JPC,nn	IN A,(n)	CC,nn	@	SBC n	RST 24
E	RET PE	JP (HL)	JPPE,nn	EX DE,HL	CPE,nn	‡	XOR n	RST 40
F	RET N	LD SP,HL	JPN,nn	EI	CN,nn	@	CP n	RST 56

@, DD or FD preceding underlined codes, exchanges the operand IX or IY respectively, for HL. In both cases the displacement, implicit in an indexed operation, follows the code.

*,‡, CB and ED precede codes shown below.

Op-codes preceded by CB

Second

First	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	RLC B	RLC C	RLC D	RLC E	RLC H	RLC L	RLC(HL)	RLC A	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC(HL)	RRC A
1	RL B	RL C	RL D	RL E	RL H	RL L	RL(HL)	RL A	RR B	RR C	RR D	RR E	RR H	RR L	RR(HL)	RR A
2	SLA B	SLA C	SLA D	SLA E	SLA H	SLA L	SLA(HL)	SLA A	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA(HL)	SRA A
3									SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL(HL)	SRL A

Bit tests, 01xx xyyy (binary)
Reset bit, 10xx xyyy (binary) where xxx is the bit number, yyy the register code
Set bit, 11xx xyyy (binary)

B—0 H—4
C—1 L—5
D—2 (HL)—6
E—3 A—7

Op-codes preceded by ED

	0	1	2	3	5	6	7	8	9	A	B	D	E	F
4	IN B,(C)	OUT(C),B	SBC HL BC	LD BC,(nn)	RET N	IM O	LDI,A	IN C,(C)	OUT(C),C	ADC HL BC	LD(nn),BC	RET I		LD R,A
5	IN D,(C)	OUT(C),D	SBC HL DE	LD DE,(nn)		IM 1	LD A,I	IN E,(C)	OUT(C),E	ADC HL DE	LD(nn),DE		IM2	LD A,R
6	IN H,(C)	OUT(C),H	SBC HL HL				RRD	IN L,(C)	OUT(C),L	ADC HL HL				RLD
7			SBC HL SP	LD SP,(nn)				IN A,(C)	OUT(C),A	ADC HL SP	DL(nn),SP			
A	LDI	CPI	INI	OUTI				LDD	CPD	IND	OUTD			
B	LDIR	CPIR	INIR	OUTIR				LDDR	CPDR	INDR	OUTDR			

To find the mnemonic corresponding to a particular hex op-code, look for the first digit of the byte down the side of the tables and for the second digit across the top.

To find the op-code corresponding to a particular mnemonic, reverse this process.

quite easy. As an example, Table 15 shows an analysis of the first part of the BURP monitor starting at address 0800. There are many subroutines in the computer's operating system and these are useful when low level programs are being written. Table 16 lists the subroutines with their CALL addresses, mnemonics and a brief description of their functions.

Development and use of machine code programs is generally a matter of personal requirement and therefore the demonstration programs will probably be of little practical use to constructors. One, however, listed in Table 17, which might be of interest to other teachers, shows the results when quanta of energy are randomly swopped between 2048 atoms (as used in Nuffield A level physics). To generate the pseudo random numbers, a 17-bit shift register with its input being the exclusive OR of the 16th and 17th bits, is set up in the Z80. There are two versions, RUN 1CCD gives a display of the atomic matrix up-dated every 256 swops and RUN 1C00 does the same, but also totals, in decimal, the number of sites with one quantum, with two quanta etc. Modifying the byte 1C04 from 31 to 32 or 33 alters the initial filling up of the matrix from all ones to all twos or threes respectively.

Table 15 Operation of part of the BURP monitor.

Hex bytes	Mnemonic	Operation performed
31 DF 1F	LD SP,1FDF	Loads the Z80 stack pointer with 1FDF
11 00 80	LD DE,8000	Loads the 16-bit register pair, DE, with 8000, which is the address of the top left-hand corner of the v.d.u.
CD CE 03	CALL 03CE	Calls the subroutine at 03CE (see Table)
20 02 02...1D		Data for the preceding subroutine (displays 'BURP')
E7	RST 32	A special one-byte CALL to a subroutine at 0020, its effect is to print a space
E7	RST 32	As above
D5	PUSH DE	Stores DE in a section of the r/w.m., using the stack pointer as a pointer to, and a reminder of, this storage 'stack'
CD C4 03	CALL 03C4	Calls the subroutine at 03C4 (clears the rest of the v.d.u. top line)
D1	POP DE	Restores the stored value of DE to the DE register pair
3E 04	LD A,04	Loads register A with the byte 04
32 EO1F	LD (1FEO),A	Loads memory location 1FEO with the contents of register A
1E 08	LD E,08	Loads register E with 08. Screen address is set eight spaces in on the top line, ready for a command

Table 16

```
*****SUBROUTINES IN MACHINE CODE*****

0254 LEAD   PROVIDES LEADER FOR TAPE.
0260 TCHAR  RECORDS (A) ON TAPE AT 300 BAUD
027F PADD   (HL) + A SPACE TO TTY
028E AHEX   CONVERTS 4 BIT HEX TO 'ASCII'
029D ASCII  'ASCII' TO TTY CONVERTER
02A5 PHEX   INSERTS SHIFTS AND SENDS TO TTY
02CC        LOOK-UP TABLE FOR TTY
02EC PSPA   TYPES A SPACE
02F0 PNEW   CARRIAGE RETURN, LINE FEED + FIG SHIFT
0301 PCHAR  UART FOR TTY
0317 LIST   LIST SUBROUTINE
0336 TIME   TIME DELAY, FOLLOWED BY LOP COUNT
0345 TCLR   CLEARS TOP LINE AND SETS TO 8000
034E 8SPA   ROUNDS SCREEN ADDRESS UP TO XXX0 OR XXX8
0355 INWRD  INPUTS AND ENCODES KEYBOARD (LAST - FIRST)
0372 MSPA   SPACER USED IN LIST AND LOAD
0393 CLR    CLEARS THE SCREEN
039F DADD   (HL) + A SPACE TO VDU
03A9 DHEX   DISPLAY A HEX BYTE ON THE VDU
            (03A9 IF THE BYTE IS IN (HL), 03AA IF IN (A))
03C4 ELIN   CLEARS OFF REST OF CURRENT VDU LINE
03CE DLIST  DISPLAY THE FOLLOWING DATA
03DE LADD   LOADS HL FROM KEYBOARD
03E7 INHEX  READS IN AND FORMS HEX BYTE
03F6 IN     READS KEYBOARD AND CONVERTS TO 4 BIT HEX
```

Table 17 Demonstration machine code program which shows the results when quanta of energy are randomly swopped between 2048 atoms.

```
1C00 21 00 0C 36   31 23 7C FE   14 20 F8 D9   21 80 0C 11
1C10 80 80 7E 12   13 23 00 7C   FE 14 20 F6   11 00 80 3E
1C20 30 06 0E C5   F5 21 00 0C   01 00 00 F1   F5 BE 20 01
1C30 03 23 7C FE   14 20 F4 C5   E1 C5 D5 11   30 30 01 E8
1C40 03 CD 82 1C   53 1E 30 01   64 00 CD 82   1C 06 30 7D
1C50 FE 0A 38 05   04 D6 0A 18   F7 E1 72 23   73 23 70 23
1C60 C6 30 77 23   EB C1 CD 4E   03 F1 C1 3C   10 B5 D9 06
1C70 00 CD 97 1C   7E FE 30 28   F8 35 CD 97   1C 34 10 F1
1C80 18 89 A7 E5   ED 42 E1 D8   ED 42 1C 18   F5 10 F1 18
1C90 B3 FF FF FF   FF FF FF CB   A9 7A E6 C0   EA A1 1C CB
1CA0 E9 CB 21 CB   13 CB 12 D5   E1 7C E6 07   C6 0C 67 C9
1CB0 21 00 0C 36   31 23 7C FE   1C 20 F8 06   00 CD 97 1C
1CC0 7E FE 30 28   F8 35 CD 97   1C 34 10 F1   D9 21 00 CC
1CD0 11 00 80 7E   12 13 23 00   7C FE 14 20   F6 D9 18 DB
1CE0 FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
```

Tape interface

The tape commands operate in the low level language, therefore, if a high level language program is to be recorded, its final address must be noted from a high level LIST. When recording it is worth spacing the blocks of recorded data because a 2 kilobyte block only requires 45 seconds of tape, and individual blocks are then easier to find. The leader of stop bits recorded automatically at the start of each recording lasts for about four seconds, so, when a recording is to be read into the computer, cue the tape just into this leader, type READ XXX, i.e. the first three characters of the hex address, start the tape and then type the last digit of the address.

In the kit of parts available for this design, one of the panel l.e.ds monitors the data stream and is turned on by the stop bits to indicate by flickering that data is being read in and, by steady illumination, that the recording has finished.

The TAPE command leaves the recording tone in the stop state so that, after the four second trailer, when the computer returns to the READY state the tone is left in the correct state for the next recording. When this trailer is reached during a READ, the computer must be interrupted by pressing a key.

Although the receiver is fairly flexible about frequencies and gives a 1 or 0, depending upon which side of 2kHz the tone is, the input from the tape recorder should be at least IV r.m.s. For recording, the output variable resistor should be set so that, without overloading the input of the tape recorder, it is possible to over-record by a few dB, quantity rather than quality being the main criterion. There is no fine adjustment of the generated frequencies because of the flexibility of the receiver design. Several different interfaces and tape decks have been tried, but a consistent error rate has been impossible to establish, even with a judiciously placed finger slowing down the tape transport.

To be continued