Appendix A: SNARK–An Abstract Teaching Machine

INTRODUCTION

This abstract machine is designed for use in an elementary computer science course. It aims to teach the elements of machine code programming, and is implemented in BASIC on a PET computer with at least 8K of writable store.

THE MACHINE

The machine consists of a CPU with two 16-bit accumulators, a 9-bit program counter, and a memory with a maximum of 512 words of immediate access store. In the present version only 128 words are available.

The CPU has a repertory of 16 instructions, of which 4 are addressless. The general format is shown below

FUNCTION	ACCUMULATOR	MODE	ADDRESS
4 bits	1 bit	2 bits	9 bits

Each instruction specifies an accumulator (either A or B), and all instructions with addresses also indicate operands.

The various functions, with their mnemonics, are as follows.

a: Functions with Addresses

LDA	Load operand to accumulator
STA	Store contents of accumulator
ADD	Add operand to accumulator
SUB	Subtract operand from accumulator
AND	Logical 'AND' operand with accumulator
ORA	Logical 'OR' operand with accumulator
JMP	Jump unconditionally to stated address
BZE	Jump if selected accumulator = 0
BNZ	Jump if selected accumulator # 0
BMI	Jump if selected accumulator < 0
BPL	Jump if selected accumulator $\geqslant 0$

b: Functions without Addresses

LRS	Shift accumulator one place right logically
NEG	Negate accumulator
INA	Input a number from the keyboard and put it into the selected accumulator
OUT	Print the number in the selected accumulator, on a line by itself
END	Stop

The mode field has four possible values

00 :	Immediate mode. The address part is taken as the operand
01 :	Direct mode. The operand is taken from the store location specified by the address
10 :	Index by accumulator A. The operand is taken from the cell pointed to by the *sum* of the address part and the current contents of accumulator A.
11 :	Index by accumulator B. The description is similar to the one above, except that accumulator B is used to calculate the address.

Note that STA and the jump instructions do not allow the immediate mode.

THE ASSEMBLY LANGUAGE

SNARK programs are entered from the keyboard using the following syntax

⟨CR⟩ ::= 'return' key
⟨digit⟩ ::= 0|1|2|3|4|5|6|7|8|9
⟨number⟩ ::= ⟨digit⟩|⟨number⟩⟨digit⟩
    e.g. : 5, 47, 123
⟨address⟩ ::= ⟨number⟩|#⟨number⟩|⟨number⟩A|⟨number⟩B
    e.g. : 14, #0, 51A, 47B

These examples illustrate the direct, immediate, index A and index B modes respectively.

⟨acc⟩ ::= A/B/⟨empty⟩
    e.g. A, B

If ⟨acc⟩ = ⟨empty⟩, then A is implied by default

⟨addressed function⟩ ::= LDA|STA|ADD|SUB|ORA|AND|JMP|BZE|BNZ|BMI|BPL
⟨addressless function⟩ ::= LRS|NEG|INA|OUT|END
⟨item⟩ ::= +⟨number⟩|–⟨number⟩|⟨addressless function⟩⟨acc⟩|
    ⟨addressed function⟩⟨acc⟩⟨address⟩
  e.g. : +4, –15, NEG B, LDA A 74A

⟨comment⟩ ::= ↑⟨followed by a sequence of any characters⟩|⟨empty⟩
⟨return⟩ ::= ⟨comment⟩⟨CR⟩
⟨labelled item⟩ ::= ⟨number⟩⟨item⟩⟨return⟩

```
e.g. :  0      +44
        1      -123     ↑MINUS ONE TWO THREE
       46      ADD B #7
```

⟨c line⟩ ::= C⟨number⟩⟨followed by a sequence of any character⟩⟨CR⟩
⟨preamble⟩ ::= ⟨preamble⟩⟨c line⟩|⟨empty⟩
⟨text⟩ ::= ⟨labelled item⟩|⟨text⟩⟨labelled item⟩
⟨program⟩ ::= ⟨preamble⟩⟨text⟩

An example of a ⟨program⟩ is shown below.

```
C0   PROGRAM TO READ, SORT AND PRINT A SET OF N
C1   NUMBERS, PRECEDED BY N
C2   STORAGE ALLOCATION:
C3      N IN 49
C4      FLAG IN 48
C5      NUMBERS IN 50 ONWARDS
 0   INA  A
 1   STA  A  49                 ↑ STORE N IN 49
 2   BZE  A   7                 ↑ JUMP IF ALL NOS READ
 3   INA  B                     ↑ READ NEXT
 4   STA  B  49  A              ↑ STORE
 5   SUB  A  #1                 ↑ DECREMENT A
 6   JMP  2                     ↑ JUMP BACK
 7   LDA  A  #0                 ↑ SET FLAG = 0
 8   STA  A  48
 9   LDA  A  #2                 ↑ SET COUNT
10   LDA  B  49  A              ↑ COMPARE PAIRS OF VALUES
11   SUB  B  48  A
12   BPL  B  20
13   STA  B  48                 ↑ SWAP IF NEED BE
14   LDA  B  49A
15   STA  B  47
16   LDA  B  48A
17   STA  B  49A
18   LDA  B  47
19   STA  B  48A
20   SUB  A  49                 ↑ SEE IF ARRAY EXHAUSTED
21   BZE  A  25
22   ADD  A  49
23   ADD  A  #1                 ↑ IF NOT, ADD 1 AND JUMP BACK
24   JMP  10
```

```
25   LDA  A  48            ↑ GO BACK IF FLAG #0
26   BNZ  A  7
27   LDA  A  49            ↑ OUTPUT RESULTS
28   LDA  B  49A
29   OUT  B
30   SUB  A  #1
31   BNZ  A  28
32   END
```

PRACTICAL DETAILS

The SNARK simulator can be loaded into the PET from a cassette tape. When started, the system will accept comment lines, program lines or any of a number of 'directives' which control the SNARK operating system.

(1) *Comment lines* A comment line starts with a C followed by a number in the range 1 to 20 and some arbitrary text. Comment lines are stored in the sequence of their numbers irrespective of the order in which they are typed. A comment line with a particular number will replace a previously entered comment with the same number.

(2) *Program lines* A program line consists of a labelled item, as defined in the previous section. Each item is sent to the SNARK store cell whose address is given at the start of the program line, so that, for example, the item in the program line

```
7 LDA B #46
```

is sent to cell 7.

A program line with a particular number will replace a previously entered line with the same number. A line can be deleted by typing its number and nothing else.

Note that program lines will generally have consecutive numbers. Unlike BASIC, new program lines may not in general be inserted between existing ones without rewriting the entire program. This is because the numbers represent *addresses*, not labels as in BASIC.

(3) *DUMP* This command will write all the material (comment and program lines) typed so far on to a cassette tape, whence it may be retrieved later by a LOAD directive.

(4) *LOAD* The LOAD directive will read a SNARK program back from a cassette tape. As it is read, the program is listed.

(5) *WIPE* All comment and program lines are deleted.

(6) *LISTALL* All comments and program lines are displayed.

(7) *LC* All comment lines are listed and displayed.

(8) *LIST n–m* m and n are integers in the range 0 to 127. Program lines n to m (inclusive) are listed.

(9) *RUN* The present program is translated from its source into binary and run. The details of the run are elicited by a brief dialogue.

(a) DISPLAY ON? If the reply is 'YES', (or Y) then each instruction and its results are displayed as the instruction is obeyed.

(b) SINGLE SHOT? If the reply is 'YES', the program is executed one instruction at a time. Each instruction is initiated by typing any key except 'B'.

(c) START ADDRESS? The reply (which should be an integer) indicates the address at which execution should start.

Execution can be stopped at any time by typing 'B' (for 'break').

The translation runs at about 2 program lines per second. It is omitted if the source program has not been altered since the last translation.

Provided that no translation has taken place, a program can safely be restarted after a break by specifying the start address '−1'.

(10) *TRANSLATE* This directive forces the translation process. It is useful in circumstances where a new translation is necessary but would have been omitted by the RUN directive. A possible case in point would be the initialisation of a self-modifying program.

(11) *PM m−n* This directive generates a post-mortem dump of registers *m−n* of the SNARK store. Each cell is displayed (a) as an instruction, and (b) as a decimal number.

PRACTICAL HINTS

(1) Cursor editing is *not* available.

(2) Under certain rare conditions, pressing the return key by itself can lead to a jump back to the BASIC monitor. The machine types

```
READY.
```

It is now rather easy to corrupt both the SNARK program *and* the underlying simulator. The only correct response is

```
GOTO 100
```

If, at this stage the user does something else, he should reload the simulator from its cassette and restart the session.

Appendix B: The SNARK Simulator Program

```
100 REM  COPYRIGHT C ANDREW COLIN 1978
110 REM SNARK SIMULATOR FOR COMMODORE PET
120 REM  ******************
130 REM  CP MARKS NEED TO RECOMPILE
140 REM  EF IS NUMBER OF ERRORS IN SOURCE
150 REM  GH IS SET=1 IF "END" OBEYED
160 REM  SS MARKS "SINGLE-SHOT"
170 REM  PC IS CONTROL COUNTER
180 REM   ******************
190 DIMC$(20)
200 DIMT$(127)
210 DIMT(127)
220 DIM M$(16)
230 CP=1
240 REM  ******************
250 REM READ MNEMONICS
260 FORJ=0TO15:READM$(J):NEXT
270 DATALDA,STA,ADD,SUB,AND,ORA,JMP,BZE
280 DATABNZ,BMI,BPL,LRS,NEG,INA,OUT,END
290 REM  ******************
300 REM  READ CODES TO BE IGNORED
310 FORJ=1TO10
320 DATA18,146,141,19,147,17,145,29,157,148
330 REM  ******************
340 REM  MAIN CONTROL POINT
350 GOSUB 1040
360 REM  LC IS "LIST COMMENTS"
370 IFX$="LC"THEN720
380 IFX$="LISTALL"THEN820
390 REM  TEST FOR "LIST A - B "
400 IF LEFT$(X$,4)="LIST" THEN 740
410 IFX$="RUN"THEN 510
420 IFX$="WIPE"THEN 860
430 IF X$="DUMP" THEN       1550
440 IF X$="LOAD" THEN       1720
450 IFX$="TRANSLATE"THEN910
460 IF LEFT$(X$,2)="PM"THEN 960
470 PRINT"DIRECTIVE NOT RECOGNISABLE"
480 GOTO350
490 REM  ******************
500 REM  RUN SEQUENCE
510 IF CP=0THEN 550
520 PRINT"TRANSLATION NEEDED.":GOSUB2370
```

```
530 IFEF=0THENPRINT"COMPILATION CORRECT":CP=0:GOTO550
540 PRINTEF;"ERRORS IN TRANSLATION":GOTO350
550 INPUT"DISPLAY ON  ";Q$:LF=1:SS=0
560 IFLEFT$(Q$,1)="N"THEN LF=0:GOTO590
570 INPUT"SINGLE SHOT";Q$
580 IFLEFT$(Q$,1)="Y"THENSS=1
590 INPUT"START ADDRESS";PX
600 IF PX=(-1)THEN 630
610 PC=PX
620 IF PC<0ORPC>127THEN590
630 GH=0
640 GOSUB2630
650 IF GH=1THEN 350
660 IFSS=1THEN690
670 GETQ$:IFQ$<>"B"THEN640
680 GOTO350
690 GETQ$:IFQ$=""THEN690
700 IFQ$="B"THEN350
710 GOTO640
720 GOSUB 1390: GOTO350
730 REM  ******************
740 X$=MID$(X$,5):GOSUB3210:REM GET PARAMETERS FOR LIST A-B
750 IF AB=1THEN780
760 GOSUB 1470
770 GOTO350
780 PRINT"LIST PARAMETERS WRONG"
790 GOTO 350
800 REM  ******************
810 REM  "LISTALL"
820 GOSUB 1390:A=0:B=127:GOSUB1470
830 GOTO350
840 REM  ******************
850 REM  "WIPE"
860 FORJ=0TO20:C$(J)="":NEXTJ
870 FORJ=0TO127:T$(J)="":NEXTJ
880 GOTO350
890 REM  ******************
900 REM  "TRANSLATE"
910 GOSUB2370
920 IFEF=0THENPRINT"COMPILATION CORRECT":CP=0: GOTO350
930 PRINTEF;" ERRORS":GOTO350
940 REM  ******************
950 REM  "PM"
960 X$=MID$(X$,3):GOSUB3210
970 IF AB=1THEN  1000
980 GOSUB3330
990 GOTO350
1000 PRINT"POST-MORTEM PARAMETERS WRONG
1010 GOTO350
1020 REM  ******************
1030 REM  READ SOURCE FROM KEYBOARD
1040 PRINT"?";:X$=""
1050 GETY$:IFY$=""THEN1050
1060 K=ASC(Y$)
1070 FORJ=1TO10
1080 IFK=C(J)THEN1050
1090 NEXT
```

```
1100 IFK<>20THEN1130
1110 IFX$=""THEN1050
1120 X$=MID$(X$,1,LEN(X$)-1):PRINT"▌ ▌";:GOTO1050
1130 PRINTY$;:IFK=13THEN1150
1140 X$=X$+Y$:GOTO1050
1150 IF X$=""THEN 1040
1160 IF LEFT$(X$,1)<>"C" THEN 1230
1170 Z=VAL(MID$(X$,2))
1180 IF Z>=0 AND Z<=20 THEN 1210
1190 PRINT"WRONG COMMENT NUMBER.RANGE IS 1-20"
1200 GOTO1040
1210 C$(Z)=X$
1220 GOTO1040
1230 Z$=LEFT$(X$,1)
1240 IF ASC(Z$)>57 OR ASC(Z$)<48 THEN 1360
1250 Z=VAL(X$)
1260 IF Z>=0 AND Z<=127 THEN 1300
1270 PRINT   "WRONG DESTINATION ADDRESS.
1280 PRINT"   RANGE IS 0 TO 127"
1290 GOTO1040
1300 X$=MID$(X$,2,LEN(X$)-1)
1310 IF X$="" THEN 1340
1320 IF LEFT$(X$,1)=" "THEN 1300
1330 IF ASC(X$)<=57AND ASC(X$)>=48THEN1300
1340 T$(Z)=X$:CP=1
1350 GOTO1040
1360 RETURN
1370 REM ******************
1380 REM LIST COMMENTS
1390 FORJ=1TO20
1400 IFC$(J)=""THEN1420
1410 PRINTC$(J)
1420 NEXTJ
1430 PRINT
1440 RETURN
1450 REM ******************
1460 REM LIST SOURCE TEXT
1470 FOR J=A TO B
1480 IFT$(J)=""THEN1500
1490 PRINT J;T$(J)
1500 NEXTJ
1510 PRINT
1520 RETURN
1530 REM ******************
1540 REM "DUMP"
1550 GOSUB 1910
1560 OPEN 1,1,1
1570 FORJ=1TO20
1580 IF C$(J)=""THEN 1610
1590 PRINT#1,C$(J)
1600 GOTO1620
1610 PRINT#1,"X"
1620 NEXTJ
1630 FOR J=0TO127
1640 IFT$(J)="" THEN 1660
1650 PRINT#1,T$(J):GOTO1670
1660 PRINT#1,"X"
```

```
1670 NEXTJ
1680 CLOSE 1
1690 GOTO350
1700 REM ******************
1710 REM "LOAD"
1720 GOSUB1910
1730 OPEN1
1740 CP=1
1750 FORJ=1TO20
1760 INPUT#1,C$(J)
1770 IF C$(J)="X" THEN 1790
1780 PRINTC$(J): GOTO 1800
1790 C$(J)=""
1800 NEXT
1810 FORJ=0TO127
1820 INPUT#1,T$(J)
1830 IF T$(J)="X" THEN 1850
1840 PRINT J;T$(J):GOTO1860
1850 T$(J)=""
1860 NEXT
1870 CLOSE 1
1880 GOTO 350
1890 REM ******************
1900 REM OPEN CASSETTE FOR READING OR WRITING
1910 PRINT"REWIND TAPE AND PRESS A KEY"
1920 GETX$:IFX$=""THEN1920
1930 RETURN
1940 REM  ******************
1950 REM COLLAPSE X$ INTO Y$
1960 Y$=""
1970 FORJ=1TOLEN(X$)
1980 Z$=MID$(X$,J,1)
1990 IF Z$="↑"THEN RETURN
2000 IFZ$=" "THEN2020
2010 Y$=Y$+Z$
2020 NEXT
2030 RETURN
2040 REM  ******************
2050 REM DECODE INSTRUCTION IN Y$
2060 E%=0:M%=0:D%=0
2070 IFLEN(Y$)<3THEN E%=1:RETURN
2080 FORJ=0TO15
2090 IFLEFT$(Y$,3)=M$(J)THEN2120
2100 NEXT
2110 E%=2:RETURN
2120 F%=J
2130 Y$=MID$(Y$,4)
2140 A%=0
2150 IF J>=11 AND LEN(Y$)=0THEN RETURN
2160 IF J<=10THEN 2200
2170 IF LEFT$(Y$,1)="A"THEN RETURN
2180 IF LEFT$(Y$,1)<>"B"THEN E%=3:RETURN
2190 A%=1:RETURN
2200 REM ADDRESSED INSTRUCTION
2210 IF LEN(Y$)=0THEN E%=4:RETURN
2220 IF LEFT$(Y$,1)="A"THEN2250
2230 IF LEFT$(Y$,1)<>"B"THEN2270
```

```
2240 A%=1
2250 Y$=MID$(Y$,2)
2260 IF LEN(Y$)=0THEN E%=4:RETURN
2270 M%=1
2280 IF LEFT$(Y$,1)<>"#"THEN 2310
2290 M%=0
2300 Y$=MID$(Y$,2)
2310 D%=ASC(Y$)
2320 IF D%<48ORD%>57THEN E%=4:RETURN
2330 D%=VAL(Y$)
2340 IF RIGHT$(Y$,1)="A"THEN M%=2
2350 IF RIGHT$(Y$,1)="B"THENM%=3
2360 RETURN
2370 REM INSTRUCTION TRANSLATOR
2380 EF=0
2390 FORH=0TO127
2400 IFT$(H)=""THEN2600
2410 X$=T$(H):GOSUB1950
2420 IF ASC(Y$)=43ORASC(Y$)=45THEN2570
2430 GOSUB2050
2440 IFE%<>0THEN2480
2450 T(H)=D%+512*M%+2048*A%+4096*F%
2460 IFT(H)>32767THEN T(H)=T(H)-65536
2470 GOTO2600
2480 EF=EF+1
2490 PRINT"ERROR IN LINE ";H
2500 PRINTH;T$(H)
2510 IFE%=1THENPRINT"TOO SHORT":PRINT
2520 IFE%=2THENPRINT"UNKNOWN MNEMONIC":PRINT
2530 IFE%=3THENPRINT"WRONG ACC DESIGNATION":PRINT
2540 IFE%=4THENPRINT"ADDRESS PART MISSING":PRINT
2550 IF E%=5THENPRINT"ADDRESS OR VALUE TOOLARGE":PRINT
2560 GOTO2600
2570 T=VAL(Y$)
2580 IFABS(T)>32767THENE%=5:GOTO2480
2590 T(H)=T
2600 NEXTH
2610 RETURN
2620 REM  *****************
2630 REMSINGLE INSTRUCTION CYCLE
2640 IR=T(PC):PC=PC+1
2650 D=IRAND511:M%=(IRAND1536)/512
2660 A%=IRAND2048:F=INT(IR/4096)
2670 IFF<0THEN F=F+16
2680 IFLF=0THEN2760
2690 PRINTPC-1;M$(F);:IFA%>0THENPRINT"B ";:GOTO2710
2700 PRINT" A ";
2710 IFF >10THENPRINT"      ";:GOTO2750
2720 IFM%=0THENPRINT"#";
2730 PRINTD;:IFM%=2THENPRINT"A";
2740 IFM%=3THENPRINT"B";
2750 PRINT"    ",
2760 CA=A0:IFA%=2048THENCA=A1
2770 IFF>7THEN2790
2780 ONF+1GOTO2810,2800,2820,2830,2840,2850,2860,2870
2790 ONF-7GOTO2890,2910,2930,2950,2960,2970,2980,2990
2800 GOSUB3090:T(D)=CA:GOTO3040
```

```
2810 GOSUB3150:CA=D:GOTO3000
2820 GOSUB3150:CA=CA+D:GOTO3000
2830 GOSUB3150:CA=CA-D:GOTO3000
2840 GOSUB3150:CA=CA ANDD:GOTO3000
2850 GOSUB3150:CA=CA OR D:GOTO3000
2860 GOSUB3090:PC=D:GOTO3040
2870 GOSUB3090:IFCA=0THENPC=D
2880 GOTO3040
2890 GOSUB3090:IFCA<>0THENPC=D
2900 GOTO3040
2910 GOSUB3090:IFCA<0THENPC=D
2920 GOTO3040
2930 GOSUB3090:IFCA>=0THENPC=D
2940 GOTO3040
2950 CA=INT(CA*0.5):GOTO3000
2960 CA=-CA:GOTO3000
2970 INPUTCA:GOTO3000
2980 PRINT"OUTPUT IS ";CA:RETURN
2990 GH=1:GOTO3040
3000 IFCA>32767THEN CA=CA-65536:GOTO3000
3010 IFCA<-32768THENCA=CA+65536:GOTO3010
3020 IFA%=0THENA0=CA:GOTO3040
3030 A1=CA
3040 IFLF=0THEN RETURN
3050 PRINT"A=";A0,"B=";A1
3060 RETURN
3070 REM  ****************
3080 REM  GET ADDRESS OF OPERAND
3090 IFM%=2THEND=D+A0
3100 IFM%=3THEND=D+A1
3110 IFD>=0ANDD<128THENRETURN
3120 PRINT"ADDRESS VIOLATION IN LINE";PC-1:D=127:GH=1:RETURN
3130 REM  ****************
3140 REM  GET OPERAND
3150 IFM%=2THEND=D+A0
3160 IFM%=3THEND=D+A1
3170 IFM%=0THENRETURN
3180 IFD>=0ANDD<128THEND=T(D):RETURN
3190 GOTO3120
3200 REM  ****************
3210 REM  GET PARAMETERS FOR LIST OR PM
3220 Y$=X$:AB=1
3230 IF Y$=""THEN RETURN
3240 Y$=MID$(Y$,2)
3250 IF Y$=""THEN RETURN
3260 IF ASC(Y$)<>45THEN3230
3270 Y$=MID$(Y$,2)
3280 A=VAL(X$):B=VAL(Y$):IF A<0ORA>127THEN RETURN
3290 IFB<0ORB>127THEN RETURN
3300 IF B<ATHEN RETURN
3310 AB=0:RETURN
3320 REM  ****************
3330 REM  INSTRUCTION CODE AND LISTING
3340 FORJ=ATOB
3350 PRINTJ;:IR=T(J):D%=IR AND 511
3360 M%=(IR AND1536)/512:A%=IRAND2048
3370 F%=IR/4096
```

```
 3380 IF F%<0THEN F%=F%+16
 3390 PRINTM$(F%);
 3400 IFA%>0THEN PRINT" B ";:GOTO3420
 3410 PRINT" A";
 3420 IFF%>10THENPRINT"  ";:GOTO3460
 3430 IFM%=0THENPRINT"#";
 3440 PRINTD%;:IFM%=2THENPRINT"A";
 3450 IFM%=3THEN PRINT"B";
 3460 PRINT" ",IR
 3470 NEXTJ
 3480 RETURN
READY.
```

Assignments

ASSIGNMENT 1 (Chapters 1 and 2)

Part A. Multiple Choice Questions

1. The use of an alphabet with a fixed number of letters limits the number of different ideas we can express: true / false.

2. The less expected a message is, the more information it carries: true / false.

3. When the number of possible messages from a source is fixed, the probabilities of the various messages add up to: less than 1 / exactly 1 / more than 1 / any of these.

4. Answer without using calculator or tables: $\log_2(1/7)$ is about: 0.14 / –0.5 / –2.8.

5. Answer without using calculator or tables: $\log_2(1)$ is about: –1 / 0 / 3.322.

6. In an ergodic message source the average message length is: always greater than the entropy / never less than the entropy / sometimes less than the entropy.

7. Is the following code ambiguous? If not, what is the average message length?

Message	*Probability*	*Code*
Blue	0.6	101
Black	0.1	110
Green	0.2	0
Red	0.1	111

Ambiguous / 2.4 / 2.6 / 2.85.

8. Is the following code ambiguous? If not, what is the average message length?

Message	*Probability*	*Code*
One	0.25	1
Two	0.25	10
Three	0.25	11
Four	0.25	100

Ambiguous / 0.25 / 2.0 / 2.5.

9. The binary system can be used to write down: a limited selection of numbers (like '101' or '111011') / any number whatever.

10. In two's complement notation there are exactly as many negative numbers as positive ones: true / false.

11. In two's complement notation, if you add two numbers of different sign the answer is always correct: true / false.

12. Octal and hexadecimal numbers are chiefly useful to: programmers and engineers / computers / both people and computers.

13. In an 8-bit 2's complement system, the hexadecimal representation of every negative numbers starts with 8, 9, A, B, C, D, E or F: true / false.

Part B. Problems

1. A game is played with tetrahedral (four-sided) dice. The two numbers thrown are added together, and the probability of each result is as follows

Result	2	3	4	5	6	7	8
Prob.	1/16	1/8	3/16	1/4	3/16	1/8	1/16

Calculate the entropy of this method of throwing the dice, when seen as an information source.

2. Devise a constant-length code for the dice throws in question 1.

3. The following coding scheme has been proposed for the dice throws in example 1.

Result	2	3	4	5	6	7	8
Code	000	001	01	100	1010	1011	11

Calculate the average message length.

Decode the following sequence of messages

01000110010010101011001001010001011111010100101101

4. Write down the binary numbers from 0 to 19.

5. Convert the following decimal numbers into their binary equivalents: 34, 10, 317, 296.

6. Convert the following binary numbers into decimal: 101011101, 11, 10101010, 1111101000.

7. Complete the following sums in binary

```
11010100        10101101        10101        1011010
10110101 +       1010011 +        101 x         1101 x
```

8. What size of binary number (i.e., how many bits) would you need to represent all the whole numbers between zero and one million?

9. Convert the following 8-bit two's complement numbers into decimal: 01010101, 11010100, 10000000, 10111111.

10. Convert the following decimal numbers into 8-bit binary, using the two's complement notation: 120, –5, 77, –100, 127.

11. Carry out the following additions in two's complement arithmetic, and state whether the answers are right

```
01011110        01111010        11010111        10001001
00010101 +      10001001 +      01111101 +      11101101 +
```

12. Assuming 8-bit words, what are the octal equivalents of the decimal values: 36, 100, –5, –59.

13. The hexadecimal form of certain two's complement binary numbers are given below. What are the decimal equivalents? 17, 7C, BA, FF, 99.

Part C. Open-ended Problems

1. Identify the source, the code, the medium, the destination and the time delay in each of the following information systems: A daily newspaper, A digital clock, A fire alarm system, An inscription in an Egyptian tomb, A traffic light.

2. Derive a formula which gives the *largest* number that can be stored in a binary word of n bits (assuming the smallest is zero).
What would be the largest numbers in words of: 4 bits, 6 bits, 10 bits?

ASSIGNMENT 2 (Chapters 3 and 4)

Part A. Multiple Choice Questions

1. The inverter is a NAND gate with only one input: true / false.

2. In a two-input NAND gate one input is 1 and the other is 0. The output is: 0 / 0.5 / 1.

3. In one second the number of nanoseconds is: a million / a thousand million / a million million.

4. The speed of a typical modern logic gate is 100 million operations per: second / minute / hour / day / year.

5. A logic network with four inputs needs a truth table with how many lines? 4 / 8 / 16.

6. Serial transmission of data is faster and cheaper than parallel transmission: true / false.

7. In binary addition (1 + 1 + 0) equals: 0 carry 1 / 1 carry 0 / 2 carry 0 / 0 carry 2.

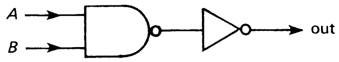
8. In a full adder, X and Y inputs can be interchanged without ill effect: true / false.

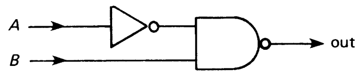
9. In a full adder, the sum and carry outputs can be interchanged without ill effect: true / false.

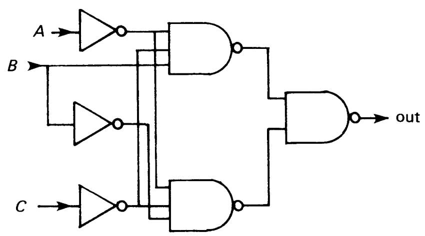
10. A multiplexer with 16 inputs needs how many control lines? 4 / 8 / 16.

Part B. Problems

1. Derive truth tables for the following networks







2. Draw networks corresponding to the following truth tables.

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

3. In a calculator each decimal digit is represented as 4 binary digits, with straightforward binary coding. For example, '3' is '0011'. The combinations 1010 to 1111 are never used.

Consider a seven-segment display, as shown in chapter 1, and complete the truth table for segment e, using '1' to mean 'this segment should light up'. The truth table begins

A	*B*	*C*	*D*	Out	
0	0	0	0	1	(since '0' uses segment e)
0	0	0	1	0	(since '1' does not use it)
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1		

Translate your truth table into a logic network.

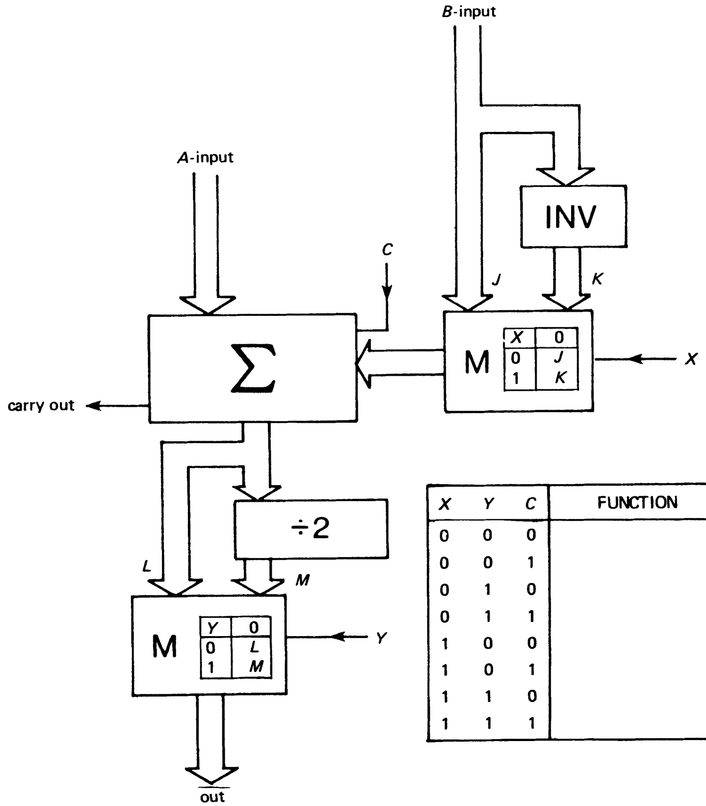
4. Some computers do not subtract by complementing and adding, but use 'full-subtractors', which are analogous to full-adders. A full-subtractor has three inputs: a digit of X, the minuend (i.e. the number being diminished), a digit of Y, the subtrahend (i.e. the number being subtracted), a 'borrow' from the previous stage. It has two outputs: a 'difference' digit (analogous to a 'sum' digit), and a 'borrow' digit (analogous to 'carry' digit). Draw up truth tables for a full subtractor, and design a logic network for the 'borrow' section only.

X	*Y*	Previous Borrow	Difference	Borrow
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

5. A binary subtractor is constructed according to the pattern shown in figure 4.5. If the numbers are 20 bits long, and the gate propagation time is 10 ns, what is the maximum propagation time of the whole subtractor? Assume that the propagation delays of a full adder are: sum digit : 3δ, carry digit : 2δ.

6. Consider the two-input multiplexer shown in figure 4.6. What is the maximum propagation delay: (a) with respect to the inputs A and B, (b) with respect to the control line X? Do these figures hold for multiplexers with greater numbers of inputs?

7. An arithmetic unit, shown below, has three control inputs: X, Y and C. Complete the table of its functions.



X	Y	C	FUNCTION
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Part C. Open-ended Problems

1. A safety device to be fitted to a car receives inputs from three sources: A = bonnet shut, B = driver's door shut, C = passenger's door shut. A safety interlock is designed so that under normal circumstances it only allows the engine to be started if the bonnet and *both* doors are shut. For testing, it also allows the engine to be started if both the bonnet and the driver's door are open – and in this case the state of the passenger's door is immaterial.

Draw up a truth table for this interlock, assuming that an output of 1 means 'it is safe to start the engine'. Show how the device could be implemented with inverters and NAND gates.

2. Most computers are fitted with 'logic' instructions, which operate on whole words, but treat each bit as an independent value (i.e., not as part of a number). The main operations are as follows

Name — *Truth Table* — *Example on 8-bit words*

NOT

A	Out
0	1
1	0

NOT 11001001
= 00110110

AND

A	*B*	Out
0	0	0
0	1	0
1	0	0
1	1	1

11001001
10100111 AND
10000001

OR

A	*B*	Out
0	0	0
0	1	1
1	0	1
1	1	1

11001001
10100111 OR
11101111

EQ

A	*B*	Out
0	0	1
0	1	0
1	0	0
1	1	1

11001001
10100111 EQ
10010001

NEQ

A	*B*	Out
0	0	0
0	1	1
1	0	1
1	1	0

11001001
10100111 NEQ
01101110

Design a 'logic unit' (by analogy with an arithmetic unit) which has two input highways for words *A* and *B*, one output highway Out and three control lines

X, Y, Z. The function of the unit is defined by the following table

X	Y	Z	FUNCTION
0	0	0	Out = A
0	0	1	Out = B
0	1	0	Out = NOT(A)
0	1	1	Out = NOT(B)
1	0	0	Out = A AND B
1	0	1	Out = A OR B
1	1	0	Out = A EQ B
1	1	1	Out = A NEQ B

Hint: Begin by designing a network for each of the functions. Then define a symbol for a network which operates on all the bits of a word at the same time, and finally connect all the networks to a parallel multiplexer.

ASSIGNMENT 3 (Chapters 5 and 6)

Part A. Multiple Choice Questions

1. The output of a flip-flop depends on its past history as well as on the current inputs: true / false.

2. When a reset signal is applied to a flip-flop in the '1' state, it changes instantaneously to the '0' state: true / false.

3. In a synchronous computer, the clock ensures that: all the flip-flops change at the same time / all the flip-flops which need to change in any one machine cycle change at the same time / the operator can tell the time of day.

4. Logic networks which pass through a number of states are called: combinatorial / sequential.

5. A register with n flip-flops can store 2^n different patterns of binary digits true / false.

6. Propagation delay: can be ignored by the computer designer / is a minor nuisance / is a major factor in computer design.

7. In a parallel machine, a 'bus' carries: 1 bit at a time / 1 word at a time / more than 1 word at a time.

8. In any one cycle, a bus allows information to be passed: from one register to one other / from one register to any number of others / simultaneously from many registers to many others.

9. In general (although there are exceptions) a given register: is always a bus master / is always a bus slave / can be either – master for one cycle, and slave the next.

10. An error in design causes two tri-state buffers driving the same bus to be on at the same time. On a particular signal line one buffer tries to transmit a '1', while the other attempts to send a '0'. What will happen? The '1' wins / The '0' wins / The signal is indeterminate and there is a risk of the machine catching fire.

11. A memory has 8K words of 8-bits each. How many bits in its MAR? 8 / 13 / 8192.

12. The MAR is always a bus slave: true / false.

13. The address selection mechanism is combinatorial: true / false.

14. Suppose you have just recorded the wrong information in a fusible-link ROM chip. Your best course of action would be: wipe the chip clear and record the information again, getting it right / send the chip back to the manufacturer for repair / throw the chip away and start with a new chip.

15. In a RAM chip the contents of all the cells can be accessed equally easily: true / false.

Part B. Problems

1. Show how a register and a parallel adder can be connected so that each clock pulse increments the contents of the register by 1 (i.e., adds 1).

2. A computer consists of nine registers arranged on a horizontal line at 10 cm intervals. Calculate the total length of highway cabling needed if (a) every register is connected to every other (i.e., cables in both directions); (b) all the registers drive a bus through a multiplexer sited in the middle of the line; (c) each register drives the bus through a tri-state buffer. (Consider only horizontal cabling.)

3. Referring to figure 5.11, tabulate the signals needed for the following transfers: ACC => MAR, ACC – DATA => ACC, 0 => ACC, 2*ACC => MAR, –1 => MAR.

Part C. Open-ended Problems

1. Write very short notes (not more than 50 words) on the following: flip-flop, D flip-flop, RAM, ROM, volatility.

2. (This question taken from University of Strathclyde, B.Sc in Computer Science, paper 52.101 for June 1979.) Describe the following components: parallel adder, parallel inverter, parallel multiplexer, register. Show how they can be connected together to form a multi-function arithmetic unit. The circuit you design should have two accumulators A and B, and be capable of the following operations

$A := B$	$B := A$
$A := (-B)$	$B := (-A)$
$A := A + B$	$B := A + B$
$A := A - B$	$B := A - B$
$A := B - A$	$B := B - A$
$A := A + 1$	$B := B + 1$

ASSIGNMENT 4 (Chapter 7)

Part A. Multiple Choice Questions

All questions refer to the SNARK computer

1. SNARK has a store with 512 cells numbered 0 upwards. The 'address' of the last one is: 511 / 512 / 65535.

2. The LDA instruction destroys the previous contents of the accumulator used: true / false.

3. The LDA instruction destroys the previous contents of the store location used: true / false.

4. What is the result, in accumulator B, of the following sequence?

```
LDA  B  #3
ADD  B  #5
SUB  B  #2
END
```

$B = 6$ / $B = 3$ / $B = 8$.

5. Two instructions are in cells 34 and 35. Can you put another instruction between them without moving at least one of the two instructions first? yes / no.

6. What are the results of the following program?

```
 0 LDA  A   35
 1 LDA  B  #17
 2 END
17 +63
35 +39
```

$A = 35, B = 17$ / $A = 35, B = 63$ / $A = 39, B = 17$ / $A = 39, B = 63$.

7. What would be the results of the following?

```
0 LDA  A   4
1 STA  A   2
2 LDA  A  #3
3 END
4 +2100
```

$A = 2100, B = 3$ / $A = 4, B = 3$ / something else.

8. What is displayed by the OUT instruction?

```
0 LDA  A  #31
1 ADD  A  #3
2 OUT  A
3 END
```

31 / 34 / something else.

9. What is displayed by the OUT instruction?

```
0 LDA  B  #47
1 LDA  A  #5
2 ADD  A   12
3 OUT  B
4 END
```

47 / 17 / something else.

10. What should be typed to make the machine print '+102'?

```
0 INA  A
1 STA  A  50
2 ADD  A  50
3 OUT  A
4 END
```

50 / 51 / 100 / 102.

Part B. Problems

Answers to problems in this section should be checked out on the SNARK simulator.

1. Write a SNARK program which calculates (361 – 153 + 7) and leaves it in accumulator *A*.

2. Write a program which calculates (1 + 2 + 3 + 4) and leaves the result in cell 2. (Hint: Do not start your program in cell 0!)

3. Write a SNARK program which accepts two numbers from the keyboard and outputs their sum (i.e., the two numbers added together).

4. Write a SNARK program which accepts three numbers from the keyboard and outputs them in the opposite order.

Part C. Open-ended Problems

Your assignment is to describe the exact operation of the AND and ORA instructions of the SNARK. Write a program which inputs two values from the keyboard, does the AND operation on them, and displays the result. Run your program with various pairs of values, keeping notes on your results, until you can explain how the results are obtained and predict the outcome of operations which you have not yet tried. Repeat this procedure for the ORA instruction. (Hint: Look at the *binary* representations of your values.)

ASSIGNMENT 5 (Chapter 8)

Part A. Multiple Choice Questions

1. The SNARK uses two's complement notation. it is correct to say that the BMI instruction jumps if (and only if) the most significant bit in the selected accumulator is 1? yes / no.

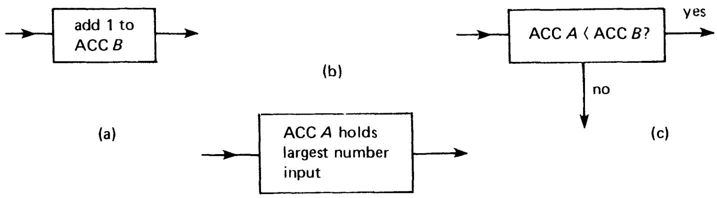
2.
```
0 INA  A
1 BZE  A 5
2 BMI  A 4
3 NEG  A
4 OUT  A
5 END
```

What will this program output if the input is 3?
3 / –3 / something else / nothing.

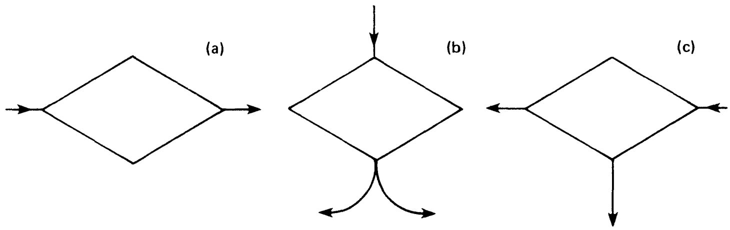
3. What will the program in question 2 output if the input is 0?
0 / something else / nothing.

4. What will the program in question 2 output if the input is –4?
4 / –4 / something else / nothing.

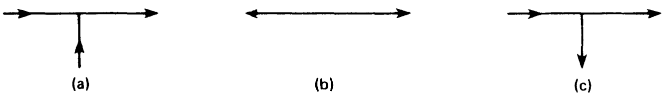
5. Which of the following could occur in a properly constructed flow chart?



6. Which of the following could occur in a properly constructed flow chart?



7. Which of the following could occur in a properly constructed flow chart?



8. In a flow chart a line links block *A* to block *B*, with the arrow pointing towards *B*. The line implies: *B* follows logically as a consequence of *A* / *B* is never started until *A* is finished / when the action specified in *A* is complete, proceed to B.

9.
```
0 LDA  A  #10
1 OUT  A
2 SUB  A  #2
3 BPL  A   1
4 END
```

How many numbers are displayed by OUT? 5 / 6 / 10.

10.
```
0 INA  A
1 STA  A 50
2 INA  A
3 OUT  A
4 SUB  A 50
5 BPL  A 3
6 END
```

Suppose that the two numbers input to this program are 5 and 37. How many numbers are displayed? 5 / 37 / 7 / 8.

Part B Problems

The 'dynamic' length of a program is the number of instructions actually obeyed when it is executed. Thus if each instruction takes one unit of time, the dynamic length is equal to the total time needed for the program to run. Give the dynamic lengths of the following programs.

```
0 LDA  A #10
1 STA  A 75
2 LDA  B #20
3 SUB  B 75
4 OUT  B
5 SUB  A #1
6 BNZ  A 1
7 END
```

```
0 INA  A
1 STA  A 80
2 INA  B
3 LDA  A #0
4 ADD  A 80
5 SUB  B #1
6 BNZ  B  4
7 OUT  A
8 END
```

(Assume that the two numbers input are 7 and 5)

```
0  INA  A
1  STA  A 50
2  INA  A
3  STA  A 51
4  LDA  A 50
5  SUB  A 51
6  BZE  A 13
7  BMI  A 10
8  STA  A 50
9  JMP  4
10 NEG  A
11 STA  A 51
12 JMP  4
13 LDA  A 50
14 OUT  A
15 END
```

(Assume that the two numbers used are 35 and 14)

2. The following program is designed to read ten numbers from the keyboard and to display their sum. Add the missing orders

```
0 LDA  A #10
1 LDA  B #0
2 STA  B  47
```

```
3  INA  B
4  ADD  B   47

5

6  SUB  A  #1

7

8  LDA  B  47
9  OUT  B
10 END
```

3. Draw a flow chart for the following program.

```
0  INA  A
1  BZE  A  10
2  LDA  B  #0
3  BMI  A  8
4  STA  A  50
5  ADD  A  50
6  ADD  B  #1
7  JMP  3
8  OUT  B
9  END
10 LDA  B  #16
11 OUT  B
12 END
```

What does this program do? (Hint: consider the *binary* form of the number used.)

Part C. Open-ended Problems

All answers should be thoroughly checked on the SNARK simulator.

1. Write a program which reads a number n and displays the sum of the first n numbers: $1+2+3+\ldots+n$.

2. Write a program which reads 10 numbers and displays the value of the largest.

3. Write a program which reads two numbers a and b, and calculates and displays (a) the quotient and (b) the remainder when a is divided by b. Use repeated subtraction. Assume both numbers are positive.

ASSIGNMENT 6 (Chapter 9)

Part A. Multiple Choice Questions

1. In the store of the SNARK, numbers and instructions are distinguished from each other because: they always occupy different 'blocks' of store / they have distinctive formats / there is no distinction; the machine obeys anything pointed to by its program counter.

2. In SNARK, every possible combination of 16 bits represents a valid instruction: true / false.

3. In SNARK, every combination of 16 bits represents an instruction with a different effect: true / false.

4. The index modes are used chiefly for manipulating arrays and tables: true / false.

5. In an index modified instruction, the contents of the modifier register is: added to the address field when the instruction is translated into machine form / added to the address field when the instruction is executed / added to the result when the instruction has been obeyed.

6. In an index modified instruction, the modifier register and the accumulator must be different: true / false.

7. Consider the sequence

```
LDA  A  #47
LDA  B  #59
ADD  B   45A
```

The effective address of the ADD instruction is: 47 / 59 / 45 / 92.

8. Consider the program

```
0  LDA  A  #3
1  LDA  B   3A
2  OUT  B
3  END
4  +4
5  +7
6  +11
7  +13
```

What is displayed? 4 / 6 / 7 / 11 / 13.

9. What area does the following sequence clear?

```
0  LDA  B  #0
1  LDA  A  #17
2  STA  B   59A
3  SUB  A  #1
4  BNZ  A   2
5  ....
```

60 – 76 / 59 – 75 / 59 – 76 / 60 – 77.

10. What is displayed?

```
0  LDA  A  #3
1  JMP      2A
2  LDA  B  #2
3  OUT  B
4  END
5  LDA  B  5
6  JMP  3
```

2 / 5 / 2 and 5 / something else.

Part B. Problems

1. Give the binary equivalents of the following SNARK instructions

```
LDA  B  #0
SUB  A  50B
BMI  B  40
OUT  A
ORA  B  77B
```

2. Give the instructions represented by the following binary values

```
1001001000010000
0111111111111111
0000000000000000
0011001100110011
0001010100000000
```

3. Translate the following numbers into SNARK instructions: –6144, +11311.

4. Translate the following SNARK instructions into denary numbers:
BMI A 35, END

5. The following program is designed to read a number in the range 0 to 3, and display '0' if the number is even, and '1' if it is odd. Fill in the missing instruction

```
0  INA  A
1
2  OUT  B
3  END
4  +0
5  +1
6  +0
7  +1
```

6. The following program is designed to display the contents of 15 cells of store-cells 74 to 60 (in that order). Fill in the spaces.

```
0  LDA
1  LDA  B
2  OUT  B
3  SUB  A  #1
4
5  END
```

7 Write a program which inputs a number in the range 1 to 15 and displays its highest prime factor. (Hint: use a table.)

8. Write a fragment of program which counts *how many* of the words in the store between cells 20 and 50 (inclusive) are different from zero. Illustrate your reply with a flow chart.

Part C. Open-ended Problems

1. Write and run a program to read 10 numbers, sort them into ascending order and display them. Use a method different from the one in appendix B, and similar in outline to the one in example 9.5. Organise a series of passes, of decreasing size, in which the largest remaining number is found and moved to the end of the table.

ASSIGNMENT 7 (Chapter 10)

Part A. Multiple Choice Questions

1. In a single cycle a machine can execute: a microinstruction / a machine instruction / neither of these.

2. Professional programmers are often expected to write microinstructions for the machines they use: true / false.

3. The 'sequence' field of a microinstruction ensures that the successor to the microinstruction is chosen correctly: true / false.

4. Microinstructions that are to be obeyed in sequence **must** normally be in consecutive locations of the ROM: true / false.

5. Every microinstruction must specify a successor: true / false.

6. Every microinstruction must specify a register transfer: true / false.

7. Any microinstruction can specify a transfer and a conditional jump: true / false

8. Any microinstruction can specify a transfer and a conditional jump which depends on the outcome of the transfer: true / false.

9. The number of machine cycles taken for the SNARK to fetch and execute the instruction ORA B 57A is: 1 / 6 / 7 / 8 /10.

10. The number of machine cycles taken to fetch and execute the instruction NEG B is: 1 / 5 / 6 / 7.

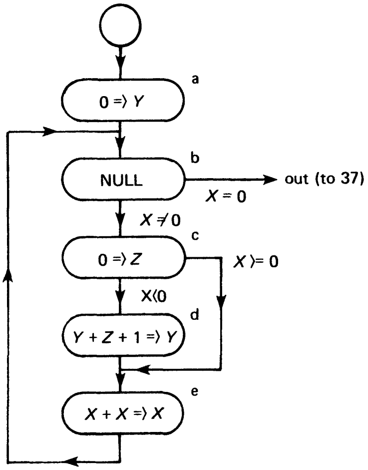
Part B. Problems

1. Using the configuration in figure 10.1, write down the shortest transfer sequence you can find to multiply the number in A by each of the following: 11, 31, 63, 64.

2. Using the configuration in figure 10.5, give a sequence of three transfers which interchanges the values in X and Y. (Hint: It can be done if you use Z as an intermediate store. Some arithmetic is necessary.)

3. The 'bit count' of a binary word is the number of 1s in it. For example, the bit count of 0011111001 is 6. Use the configuration given in example 10.2 to write a microcode sequence which generates a bit count. The sequence is to start with the word to be bit-counted in register X, and the result is to be placed in register Y. The value in X need not be preserved, and Z can be used if necessary. The first microinstruction of your sequence should be in location 14, and on completion control is to be transferred to location 37. Give your answer both in symbolic and in binary form. (Hints: Remember that the '$X < 0$' condition is true if the most significant digit of X is a 1, regardless of the other

digits. Note that when a binary number is added to itself, it is effectively shifted one place left, and the previous most significant digit disappears. Use the following microflow chart. The '0 => Z' transfer has been included since, with the given configuration, it is not possible to increment Y by 1 except by addition.)



Part C. Open-ended Problems

1. 'Exclusive OR' is a logical operation which uses two operands and generates one result. Each bit in the result is a '1' if and only if the corresponding bits in the operands are *different*. For example 00110100 ⊕ 01010110 = 01000010 (where ⊕ means 'exclusive OR'). The order code of SNARK is to be altered so that the NEG A and NEG B instructions are replaced by two new instructions

EOR A A ⊕ B => A
EOR B A ⊕ B => B

Design appropriate changes to the microcode.

ASSIGNMENT 8 (Chapters 11, 12 and 13)

Part A. Multiple Choice Questions

1. Computer memories and public libraries are both stores of information. In computing terms, is a large library: a random access device / a cyclic device / a serial device?

2. Answer question 1 for a travelling library (i.e., one which is housed in a van and visits your village at regular intervals): a random access device / a cyclic device / a serial device.

3. What is the latency of a store? The time needed to gain access to any item of information it contains / the time needed to extract and copy *all* the information it contains / the time that the information can be expected to remain stored accurately and reliably.

4. How can the information on a magnetic tape best be preserved safely? by keeping the tape in suitable air-conditioned rooms / by making sure that the tape deck mechanism is well maintained / by copying the information to another tape.

5. What is a filing system? A system for organising data on a backing store / a system for ensuring that information is preserved even though disc crashes and other accidents may occur / a system for preventing unauthorised access to information by people who have no right to it / a system which is responsible for all three of these functions.

6. The best quality of print is produced by a matrix printer: true / false.

7. In the context of document production, what is 'justification'? Getting the margins straight on both sides / getting the spelling right / marking up the text for the printer.

8. The chief advantage of a symbolic assembler is that: it simplifies transfer of programs to other machines / it simplifies alterations to programs / Programs written in symbolic assembly language use the computer more efficiently than those written in machine code.

9. A 'compiler' for a high-level language is: a machine ('hardware') / a program ('software') / a person ('liveware').

10 The prime aim of a multi-access system is: to allow its users to communicate with one another / to allow a large number of people to use the same program at the same time / to allow a large number of people to use the computer simultaneously and independently.

Part B. Problems

1. A magnetic tape store has the following characteristics: length of tape: 3600 feet; recording density: 2000 bytes per inch; reading speed: 100 inches per second; gap between records: 1 inch; time needed to start tape moving or to

stop it: 10 milliseconds. If the record size is 1000 characters, and the tape has to be stopped between each record, how much information can be stored on the tape? How long will it take to read through the entire tape? (Give your answers to the nearest megabyte and minute, respectively.)

2. Repeat question 1, assuming a block size of 10 000 characters.

Part C. Open-ended Problems

1. List the types and capacities of the various storage systems on the computer you are using for your present course.

2. Find out as much as you can about the system software on your local computer and write short notes under each of the following headings: Operating system(s); Languages available; Accounting system and resource control; Other facilities.

3. Write a short essay (1 page) on input and output. Use examples with which you are personally familiar.

ASSIGNMENT 9 (Chapter 14)

Part A. Multiple Choice Questions

1. Consider the grammar

   ⟨var⟩ ::= X|Y|Z
   ⟨exp⟩ ::= ⟨var⟩|⟨exp⟩+⟨var⟩|⟨exp⟩–⟨var⟩

Which of the following sequences is an ⟨exp⟩? –X+Y / XYZ / X+Y–Z.

2. Consider the grammar

   ⟨var⟩ ::= A|B|C|(⟨exp⟩)
   ⟨exp⟩ ::= ⟨var⟩|–⟨var⟩|⟨exp⟩+⟨var⟩|⟨exp⟩*⟨var⟩

Which sequence is an ⟨exp⟩? (A+B)*(A+(A*B)+C) / –(A+B)(A+C) / –A+–B.

3. It is easier to define the semantics of a language than its syntax: true / false.

4. Assuming the rules given on p. 160, what is the precedence of the + in '*a**(*b*+*c*)'? 1 / 2 / 3 / 4.

5. Which is the valid reverse Polish expression? (*a*+*b*) / *abc*+++ / ZC+I+W*E/I+C–AKUL+*+ / .

6. Consider the reverse Polish expression '*abc*/+'. Which diagram best describes the stack after the evaluation of the '/'?

(1)	(2)	(3)
c/*b*	*b*/*c*	1/*c*
a	*a*	*a*+*b*

7. What is the largest number of stack cells in use at any one time when the following reverse Polish expression is evaluated: '*ab*+*cde*–**x*/–': 1 / 2 / 3 / 4 / 5 / 6.

8. Which of the following expressions is equivalent to the one in question 7? *cde*–**x*/*ab*+– *ade*–*c***x*/–*b*+ *ab*+*xcde*–*/–.

9. In general, an interpreter is cheaper to produce than a compiler: true / false.

10. In general, an interpreter generates more efficient machine code than a compiler: true / false.

Part B. Problems

1. Use the SNARK grammar given in appendix B to draw parse trees for each of the following items: OUT A, BZE A 35, LDA B ≠107, +346.

2. In the following state-symbol table, ⟨digit⟩ ::= 0|1|2|3|4|5|6|7|8|9.

	STATE						
SYMBOL	1	2	3	4	5	6	7
⟨digit⟩	11 $\surd 3$	16 $\surd 3$	21 $\surd 3$	26 $\surd 4$	31 $\surd 7$	36 $\surd 7$	41 $\surd 7$
	12	17	22 $\surd 4$	27	32	37	42
+ or –	13 $\surd 2$	18	23	28	33 $\surd 6$	38	43
E	14	19	24 $\surd 5$	29 $\surd 5$	34	39	44
;	15	20	25 $\surd 0$	30 $\surd 0$	35	40	45 $\surd 0$

To help you answer the question, each cell in the table carries a *label*, which is a number in the range 11 to 45. Consider each of the following strings, and decide whether it conforms to the grammar defined by the table. Give a list of the cells visited for string, as shown in the example.

	Cells visited	Correct?
345;	11, 21, 21, 25	Yes
+35.73;		
−12.19.7;		
44.;		
−1E7;		
−3E+;		

3. Convert the following into reverse Polish using Dijkstra's algorithm: $A+B$, $X+(Y-3)$, $(A+B)*(C-D)$, $\dfrac{A+B*C}{X+2}+\dfrac{A-B/C}{X-2}$, sqrt$(X+Y-5)$. (Note that in reverse Polish operators with only one argument, like sqrt, are written after their operands.)

4. The reverse Polish equivalent of the expression $(A-B)/(C+D)$ is $AB-CD+/$. Suppose that variables A, B, C and D have been allocated cells 100, 101, 102 and 103 respectively. Write down the SNARK code obtained by compiling the above expression, using the automatic method described in chapter 14.

5. Write down the shortest code sequence you can think of to evaluate the expression in question 4. Use location 90 as a workspace if necessary.

Part C. Open-ended Problems

1. (This question taken from University of Strathclyde, B.Sc in Computer Science, paper 51,101 for June 1979.) In a certain programming language declarations are written in one of three forms

⟨identifier⟩;
or ⟨identifier⟩ := ⟨number⟩;
or [⟨number⟩] ⟨identifier⟩;

where ⟨identifier⟩ ::= ⟨identifier⟩⟨digit⟩⟨identifier⟩⟨letter⟩|⟨letter⟩
and ⟨number⟩ ::=⟨number⟩⟨digit⟩|⟨digit⟩

for example

```
x;
why := 123;
[99] zed3;
```

Assuming that the symbols in the language have been categorised into the classes letter digit := [ ] ; and 'other', draw up a state-symbol table which checks the grammar of a declaration.

ASSIGNMENT 10 (Chapters 15 to 18)

1 It is the year 2000. You have been asked to revise this textbook. Write a section to be added to chapter 15, outlining the history of computers between 1980 and 2000. A list of suggested 'keywords' for your answer is: VLSI, multi-computer systems, distributed data bases, communication, home computing, Josephson effect, automated office, robotics, artificial intelligence, ultra-reliable and self-repairing systems.

Sample Solutions

ASSIGNMENT 1

A1 false; **A2** true; **A3** exactly 1; **A4** –2.8; **A5** 0; **A6** never less than the entropy; **A7** 2.6; **A8** ambiguous; **A9** any number whatever; **A10** false; **A11** true; **A12** programmers and engineers; **A13** true

B1 2.656

B2

Result	2	3	4	5	6	7	8
Your code	000	001	010	011	100	101	110

(Any unambiguous 3-bit code will serve.)

B3 $3\frac{1}{8}$, 428334445563488 6574; **B4** 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, 10001, 10010, 10011; **B5** 100010, 1010, 100111101, 100101000; **B6** 349, 3, 170, 1000

B7 110001001 100000000 1101001 10010010010

B8 20; **B9** 85, –44, –128, –65; **B10** 01111000, 11111011, 01001101, 10011100, 01111111; **B11** 01110011 correct, 00000011 correct, 01010100 correct, 01110110 incorrect; **B12** 044, 144, 373, 305; **B13** +23, +124, –70, –1, –103

C1 reporter – English – newsprint – reader – 1 day, etc.
C2 $2^n - 1$, 15, 63, 1023

ASSIGNMENT 2

A1 true; **A2** 1; **A3** a thousand million; **A4** second; **A5** 16; **A6** false; **A7** 0 carry 1; **A8** true; **A9** false; **A10** 4

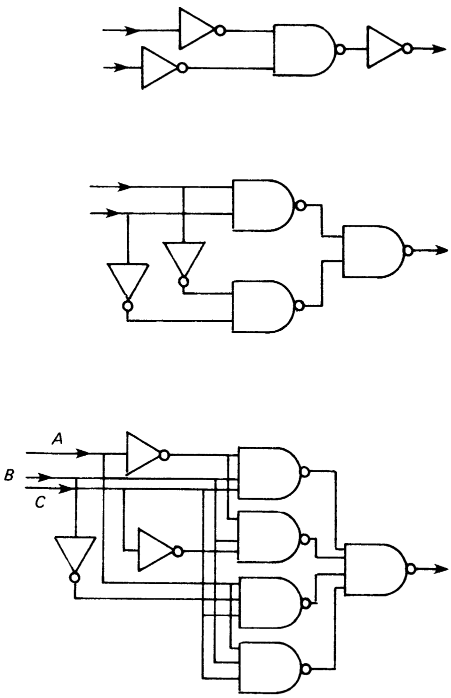
B1

A	*B*	Out
0	0	0
0	1	0
1	0	0
1	1	1

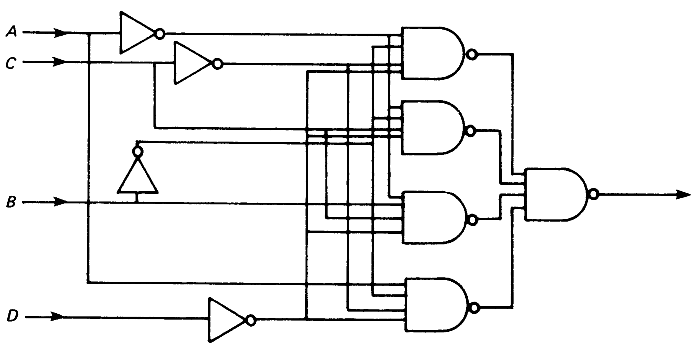
A	*B*	Out
0	0	1
0	1	0
1	0	1
1	1	1

A	*B*	*C*	Out
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

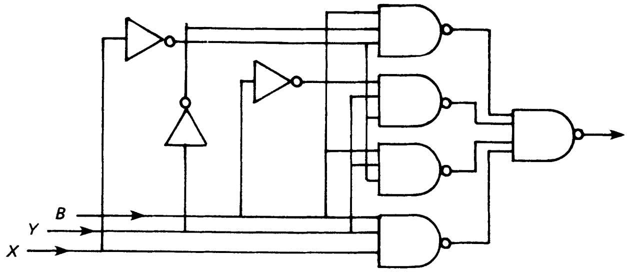
B2



B3 10001010



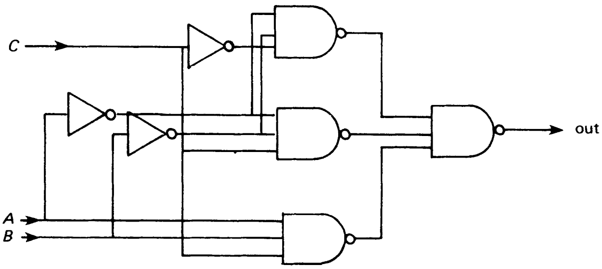
B4 difference: 01101001, borrow 01110001



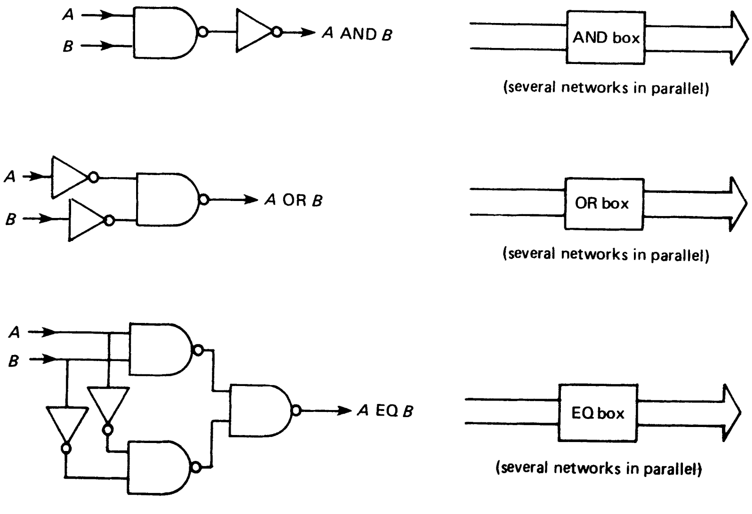
B5 410 ms; **B6** $2\delta, 3\delta$, yes; **B7** $A + B, A + B + 1, \frac{1}{2}(A + B), \frac{1}{2}(A + B + 1),$
$A - B - 1, A - B, \frac{1}{2}(A - B - 1), \frac{1}{2}(A - B)$

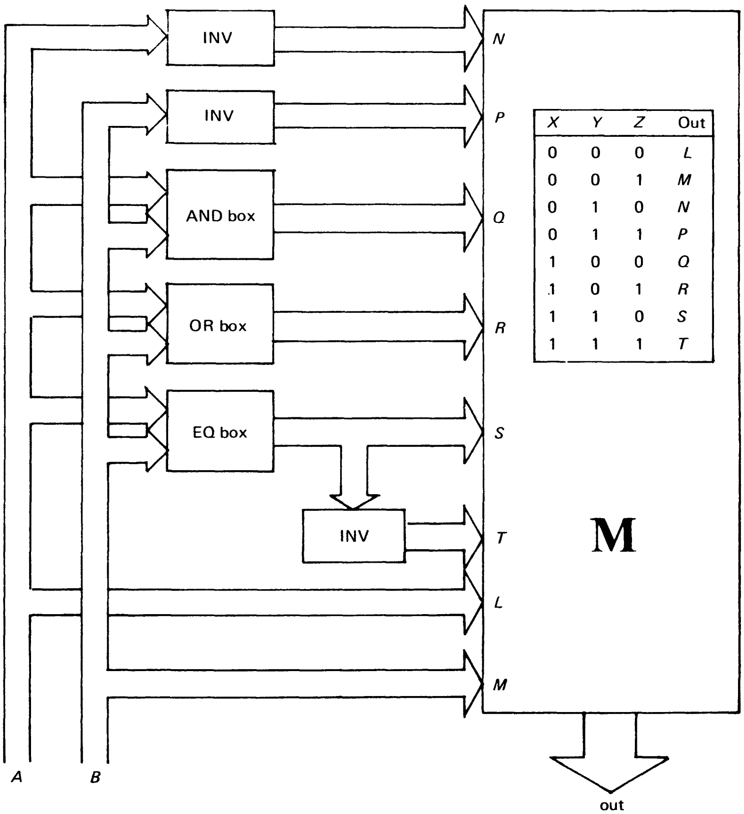
C1

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



C2

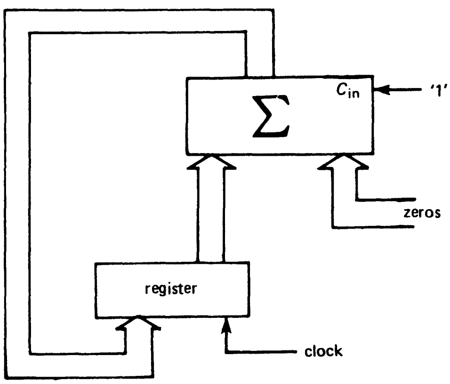




ASSIGNMENT 3

A1 true; **A2** false; **A3** all the flip-flops which need to change in any one machine cycle change at the same time; **A4** sequential; **A5** true; **A6** is a major factor in computer design; **A7** 1 word at a time; **A8** from one register to any number of others; **A9** can be either; **A10** the signal is indeterminate; **A11** 13; **A12** true; **A13** true; **A14** throw the chip away; **A15** true

B1

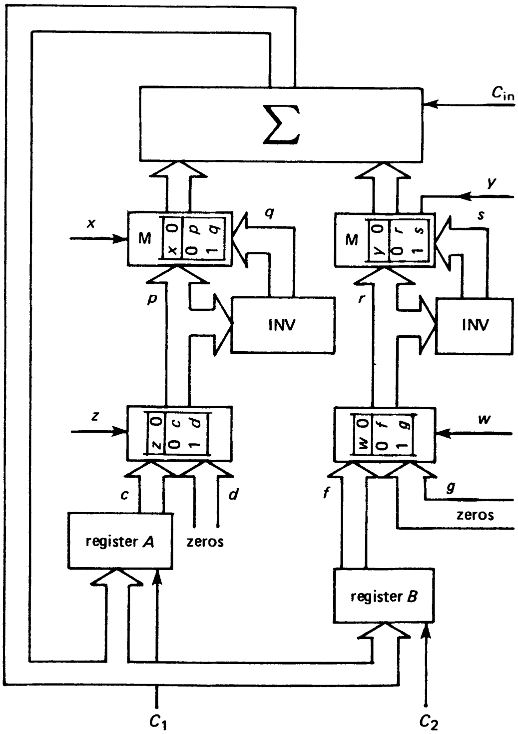


B2 2400 cm, 280 cm, 80 cm

B3

TRANSFER	X	Y	F	G	C	S_1	S_2	S_3	S_4	C_1	C_2	C_3	C_4
ACC => MAR	–	–	–	–	–	0	1	0	0	⎍			
ACC–DATA => ACC	0	1	1	0	1	1	0	0	0	⎍			
0 => ACC	0	1	0	0	1	1	0	0	0	⎍			
2*ACC => MAR	0	0	0	0	0	1	0	0	0				⎍
–1 => MAR	0	1	0	0	0	1	0	0	0				⎍
or	1	1	0	1	0	1	0	0	0				⎍

C2



	x	y	z	w	C_{in}	C_1	C_2
$A = B$	0	0	1	0	0	⎍	
$B = A$	0	0	0	1	0		⎍
$A = (-B)$	0	1	1	0	1	⎍	
$B = (-A)$	1	0	0	1	1		⎍
$A = A + B$	0	0	0	0	0	⎍	
$B = A + B$	0	0	0	0	0		⎍
$A = A - B$	0	1	0	0	1	⎍	
$B = A - B$	0	1	0	0	1		⎍
$A = B - A$	1	0	0	0	1	⎍	
$B = B - A$	1	0	0	0	1		⎍
$A = A + 1$	0	0	0	1	1	⎍	
$B = B + 1$	0	0	1	0	1		⎍

ASSIGNMENT 4

A1 511; **A2** true; **A3** false; **A4** $B = 6$; **A5** no; **A6** $A = 39, B = 17$; **A7** something else; **A8** 34; **A9** 47; **A10** 51

B1

```
0 LDA A #361
1 SUB A #153
2 ADD A #7
3 END
```

B2

```
3 LDA B #1
4 ADD B #2
5 ADD B #3
6 ADD B #4
7 STA B 2
8 END
```

B3

```
0 INA A
1 STA A 30
2 INA A
3 ADD A 30
4 OUT A
5 END
```

B4

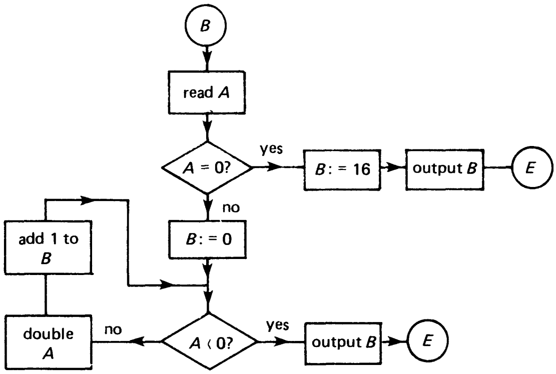
```
0 INA A
1 STA A 50
2 INA A
3 INA B
4 OUT B
5 OUT A
6 LDA A 50
7 OUT A
8 END
```

C AND does a logical 'and' independently in each pair of bits in the operands. ORA does a logical 'or' in the same way.

ASSIGNMENT 5

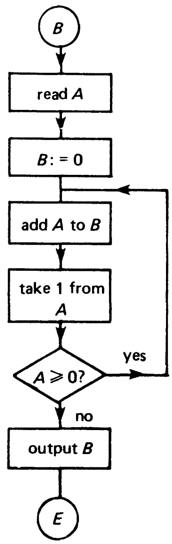
A1 yes; **A2** −3; **A3** nothing; **A4** −4; **A5** (a); **A6** (c); **A7** (a);
A8 when the action specified in *A* is complete, proceed to *B*; **A9** 6; **A10** 8

B1 62, 21, 29; **B2** STA B 47, BNZ A 3; **B3** The program displays the number of leading zeros (on the left) of the binary representation of the number used.



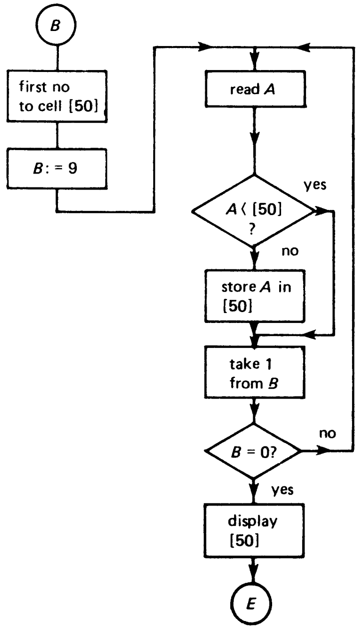
C1

```
0 INA  A
1 LDA  B  #0
2 STA  A  50
3 ADD  B  50
4 SUB  A  #1
5 BPL  A  2
6 OUT  B
7 END
```



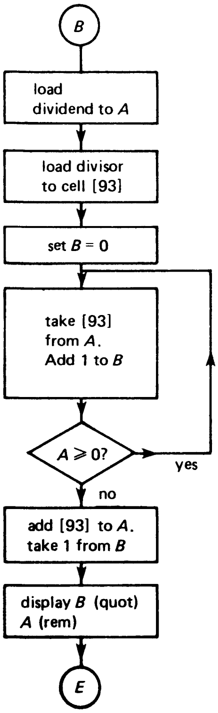
C2

```
 0  INA  ,A
 1  STA  A  50
 2  LDA  B  #9
 3  INA  A
 4  SUB  A  50
 5  BMI  A  8
 6  ADD  A  50
 7  STA  A  50
 8  SUB  B  #1
 9  BNZ  B  3
10  LDA  A  50
11  OUT  A
12  END
```



C3

```
 0  INA  A
 1  INA  B
 2  STA  B  93
 3  LDA  B  #0
 4  SUB  A  93
 5  ADD  B  #1
 6  BPL  A  4
 7  ADD  A  93
 8  SUB  B  #1
 9  OUT  B
10  OUT  A
11  END
```



ASSIGNMENT 6

A1 there is no distinction; **A2** true; **A3** false; **A4** true; **A5** added to the address field when the instruction is executed; **A6** false; **A7** 92; **A8** 11; **A9** 60 – 76; **A10** something else (+2565)

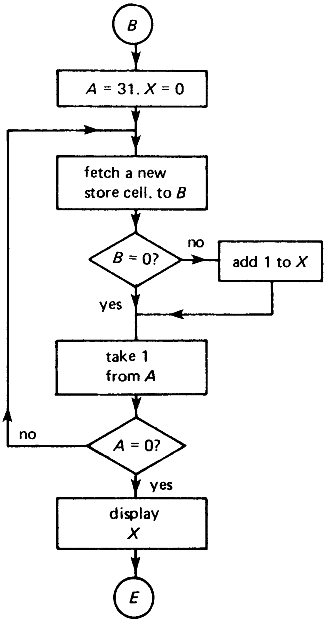
B1 0000100000000000, 0011011000110010, 1001101000101000, 110000000000000, 0101111001001101; **B2** BMI A 16, BZE B 511B, LDA A #0,SUB A 307, STA A 256A; **B3** OUT B, ADD B 47A; **B4** –28125, –4096; **B5** LDA B 4A; **B6** A #15, 59A, #, BNZ A 2;

B7

```
 0  INA  A
 1  LDA  A  3A
 2  OUT  A
 3  END
 4  +1
 5  +2
 6  +3
 7  +2
 8  +5
 9  +3
10  +7
11  +2
12  +3
13  +5
14  +11
15  +3
16  +13
17  +7
18  +5
```

B8

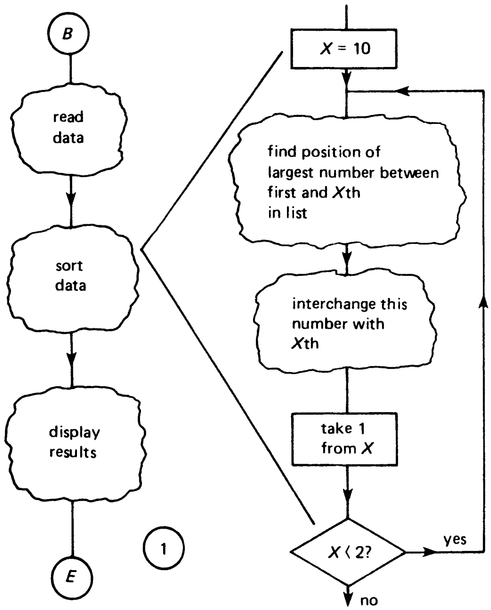
```
 0  LDA  A  #31
 1  LDA  B  #0
 2  STA  B  19
 3  LDA  B  19A
 4  BZE  B  8
 5  LDA  B  19
 6  ADD  B  #1
 7  STA  B  19
 8  SUB  A  #1
 9  BNZ  A  3
10  LDA  B  19
11  OUT  B
12  END
```



C1

```
 0  LDA  B  #10     ↑ READ 10 NOS AND PLANT IN
 1  INA  A          ↑ 70–79
 2  STA  A  69B
 3  SUB  B  #1
 4  BNZ  B  1
 5  LDA  A  #10     ↑ SET X (IN 69) = 10
 6  STA  A  69
 7  LDA  B  69      ↑ B := X
 8  LDA  A  69B
 9  STA  A  68      ↑ PUT XTH NO IN 68 (LARGEST SO FAR)
10  STA  B  67      ↑ PUT POSITION OF LARGEST IN 67
11  SUB  B  #1
12  LDA  A  69B     ↑ GET ANOTHER NO
13  SUB  A  68      ↑ COMPARE WITH LARGEST
14  BMI  A  18
15  LDA  A  69B     ↑ IF LARGER SET UP NEW VALUES
16  STA  A  68      ↑ IN 68, 67
17  STA  B  67
18  SUB  B  #1
19  BNZ  B  12      ↑ LOOP ROUND
20  LDA  B  69
```

```
21  LDA  A  69B        ↑ GET XTH NO
22  LDA  B  67
23  STA  A  69B        ↑ PUT IN PLACE OF LARGEST
24  LDA  B  69
25  LDA  A  68         ↑ GET LARGEST
26  STA  A  69B        ↑ PUT IN PLACE OF XTH
27  SUB  B  #1
28  STA  B  69
29  SUB  B  #2         ↑ LOOP ROUND
30  BPL  B  7
31  LDA  A  #10        ↑ DISPLAY RESULTS
32  LDA  B  69A
33  OUT  B
34  SUB  A  #1
35  BNZ  A  32
36  END
```



ASSIGNMENT 7

A1 a microinstruction; **A2** false; **A3** true; **A4** false; **A5** true; **A6** false; **A7** true; **A8** false; **A9** 8; **A10** 6

B1

11	31	63	64
A => B	A => B	A => B	A*2 => A
A*2 => A	A*2 => A	A*2 => A	A*2 => A
A*2 => A	A*2 => A	A*2 => A	A*2 => A
A+B => A	A*2 => A	A*2 => A	A*2 => A
A*2 => A	A*2 => A	A*2 => A	A*2 => A
A+B => A	A*2 => A	A*2 => A	A*2 => A
	A − B => A	A*2 => A	
		A − B => A	

B2 X + Y => Z, X + (0) => Y, Z − X => X

B3

		Symbolic	*Binary*					
Location	Transfer	Sequence	Address *a*	Address *b*	T_1	T_0	*JKLMN*	$S_1S_2C_1C_2C_3$
a (14)	0 => Y	goto b (15)	001111	000000	0	0	11000	1 0 0 1 0
b (15)	NULL	if X = 0 then goto 37 else c	100101	010000	0	1	00000	0 0 0 0 0
c (16)	0 => Z	if X < 0 then d else e	010010	010001	1	0	11000	1 0 0 0 1
d (17)	Y+Z+1 => Y;	goto e	010010	000000	0	0	10011	1 0 0 1 0
e (19)	X+X => X;	goto b	001111	000000	0	0	00000	1 0 1 0 0

C1 We note that $A \oplus B = (A \text{ OR } B) - (A \text{ AND } B)$. A suitable sequence for $A \oplus B \Rightarrow A$ is

A *and* B => DR
A *or* B => A
A − DR => A

and similarly for $A \oplus B \Rightarrow B$. We make the following changes

Location	*New Contents*
15	A *or* B => A; goto 38
16	A *or* B => B; goto 39
56	A *and* B => DR; goto 15
57	A *and* B => DR; goto 16

ASSIGNMENT 8

A1 a random access device; **A2** a cyclic device; **A3** the time needed to gain access; **A4** by copying the information to another tape; **A5** a system responsible for all three functions; **A6** false; **A7** getting the margins straight on both sides; **A8** it simplifies alterations to programs; **A9** a program; **A10** to allow a large number of people to use the computer simultaneously and independently

B1 29 megabytes, 12 minutes; **B2** 72 megabytes, 6 minutes

ASSIGNMENT 9

A1 X + Y − Z; **A2** (A + B)*(A + (A * B) + C); **A3** false; **A4** 3;
A5 ZC + I + W * E/I + C − AKUL + * + /; **A6** (2); **A7** 4;
A8 *ade − c*x/−b+*; **A9** true; **A10** false

B1

```
OUT A   ⟨item⟩
          |
        ⟨addressless function⟩⟨acc⟩
          |                     |
          OUT                   A
```

```
BZE A 35   ⟨item⟩
             |
           ⟨addressed function⟩⟨acc⟩⟨address⟩
             |                   |     |
             BZE                 A   ⟨number⟩
                                       |
                                     ⟨number⟩⟨digit⟩
                                       |        |
                                     ⟨digit⟩    5
                                       |
                                       3
```

```
LDA B ≠107   ⟨item⟩
              |
             ⟨addressed function⟩⟨acc⟩⟨address⟩
              |                    |     |
             LDA                   B    #⟨number⟩
                                          |
                                         ⟨number⟩⟨digit⟩
                                          |        |
                                          |        7
                                         ⟨number⟩⟨digit⟩
                                          |        |
                                         ⟨digit⟩   0
                                          |
                                          1
```

```
+346   ⟨item⟩
       +⟨number⟩
         |
        ⟨number⟩⟨digit⟩
         |        |
         |        6
        ⟨number⟩⟨digit⟩
         |        |
        ⟨digit⟩   4
         |
         3
```

B2

	Cells visited	Correct?
345;	11, 21, 21, 25	Yes
+35.73;	13, 16, 21, 22, 26, 26, 30	Yes
–12.19.7;	13, 16, 21, 22, 26, 26, 32	No
44.;	11, 21, 22, 30	Yes
–1E7;	13, 16, 24, 31, 45	Yes
–3E+;	13, 16, 24, 33, 40	No

B3 AB +, XY3 – +, AB+CD–*, ABC*+.X2+/ABC/ –X2–/+, XY+5–sqrt

B4

```
LDA  B  #0
LDA  A  100     A
STA   A  90B
ADD  B  #1
LDA  A  101     B
STA  A  90B
ADD  B  #1
SUB  B  #1      –
LDA  A  89B
SUB  A  90B
STA  A  89B
LDA  A  102     C
STA  A  90B
ADD  B  #1
LDA  A  103     D
STA  A  90B
ADD  B  #1
SUB  B  #1      +
LDA  A  89B
ADD  A  90B
STA  A  89B
SUB  B  #1      /
LDA  A  89B
DIV  A  90B
STA  A  89B
```

(25 instructions)

B5

```
LDA  A  102
ADD  A  103
STA  A   90
LDA  A  100
SUB  A  101
DIV  A   90     (6 instructions)
```

C1

	State							
Symbol	1	2	3	4	5	6	7	8
⟨letter⟩	√2	√2					√8	√8
⟨digit⟩		√2	√4	√4	√6	√6		√8
:=		√3						
[	√5							
]						√7		
;		√0		√0				√0
⟨other⟩								

Note: Order of rows and columns is not significant!

Bibliography

Barron, D. W., *Assemblers and Loaders* (MacDonald/Elsevier, 1968)

Boulaye, G. G., *Microprogramming* (Macmillan, 1975)

Calderbank, V. J., *A Course of Programming in FORTRAN IV*, (Chapman & Hall, 1969)

Colin, A. J. T., *Programming and Problem Solving in Algol 68* (Macmillan, 1975)

Commodore Ltd, *PET User Manual* (October, 1978)

Findlay, W., and Watt, D. A., *Pascal – An Introduction to Methodical Programming* (Pitman, 1978)

Hartley, M. G., and Healey, M., *A First Course in Computer Technology* (McGraw-Hill, 1978)

Jackson, M., *Principles of Program Design* (Academic Press, 1975)

Jensen, K., and Wirth, N., *Pascal-User Manual and Report* (Springer-Verlag, 1975)

Kernighan B. W., and Plauger, P. J., *Software Tools* (Addison-Wesley, 1976)

Koestler, A., *Janus – A Summing Up* (Hutchinson, 1978)

Lavin, D., *Logical Design of Switching Circuits* (Nelson, 1974)

Lister, A. M., *Fundamentals of Operating Systems,* second edition (Macmillan, 1980)

Mowle, F. J., *A Systematic Approach to Digital Logic Design* (Addison-Wesley, 1976)

Scientific American, issue on Micro-electronics (September, 1977)

The TTL Data Book (Texas Instruments, 1980)

Weinberg, G. M., *The Psychology of Computer Programming* (Van Nostrand, 1971)

Weizenbaum, J., *Computer Power and Human Reason* (Freeman, 1976)

Index