INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION

INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
34100 TRIESTE (ITALY) - P.O.B. 586 - MIRAMARE - STRADA COSTIERA 11 - TELEPHONES: 224281/2/3/4/5/6
CABLE: CENTRATOM - TELEX 460392-1

SMR/90 - 15

COLLEGE ON MICROPROCESSORS:

TECHNOLOGY AND APPLICATIONS IN PHYSICS

7 September - 2 October 1981

ARITHMETIC AND SUBROUTINES

M.D. EDWARDS

Computation Dept.
U.M.I.S.T.
Manchester
England

These are preliminary lecture notes, intended only for distribution to participants. Missing or extra copies are available from Room 230.

2's COMPLEMENT NUMBERS ①

- WHAT ARE THEY?
  - 2's COMPLEMENT NOTATION PROVIDES A MECHANISM FOR REPRESENTING POSITIVE AND NEGATIVE NUMBERS (IN BINARY FORM).
- HOW ARE THEY DERIVED?

  A BINARY NUMBER MAY BE REPRESENTED IN n BITS.
  THE 2's COMPLEMENT OF A NUMBER N MAY BE FOUND AS FOLLOWS

  - 2's COMPLEMENT OF $N = 2^n - N$ (10000 − N)

- THE RANGE OF 2's COMPLEMENT NUMBERS WHICH MAY BE REPRESENTED IN i BITS IS

$$-2^{i-1} \leq N \leq (2^{i-1} - 1)$$

- FOR EXAMPLE $n = 4$

RANGE $= -2^3 \leq N \leq (2^3 - 1)$

$= -8 \leq N \leq 7$

FOR $n = 8$ (word length of 6800)

RANGE $= -2^7 \leq N \leq (2^7 - 1)$

$= -128 \leq N \leq 127$

TABLE OF 2's COMPLEMENT NUMBERS ②

n = 4 bits

0 1 1 1	+ 7
0 1 1 0	+ 6
0 1 0 1	+ 5
0 1 0 0	+ 4
0 0 1 1	+ 3
0 0 1 0	+ 2
0 0 0 1	+ 1
0 0 0 0	0
1 1 1 1	- 1
1 1 1 0	- 2
1 1 0 1	- 3
1 1 0 0	- 4
1 0 1 1	- 5
1 0 1 0	- 6
1 0 0 1	- 7
1 0 0 0	- 8

↑ SIGN BIT

```
 1111
 1110
-----
01101
-----
```

- TO FIND THE 2's COMPLEMENT OF A NUMBER (6).

	0 1 1 0 (+6)
1. INVERT EACH BIT	1 0 0 1
2. ADD 1	1 0 1 0 (-6)

2'S COMPLEMENT ARITHMETIC

1. $5-3 \equiv 5+(-3)$

	0 1 0 1	(+5)
+	1 1 0 1	(-3)
	0 0 1 0	(+2)

2. $6+2$

	0 1 1 0	(+6)
+	0 0 1 0	(+2)
	1 0 0 0	(-8)

$-8 \leq N \leq 7$

- ANSWER IS WRONG BECAUSE +8 CANNOT BE REPRESENTED IN 4 BITS
  - THIS IS KNOWN AS **OVERFLOW**

3. $-4-5 \equiv -4+(-5)$

	1 1 0 0	(-4)
+	1 0 1 1	(-5)
	0 1 1 1	(+7)

- AGAIN ANSWER IS WRONG DUE TO OVERFLOW
- THE 6800 CAN DETECT OVERFLOW IN 2'S COMPLEMENT ARITHMETIC
  - OVERFLOW FLAG = V
- EQUATION FOR V IS

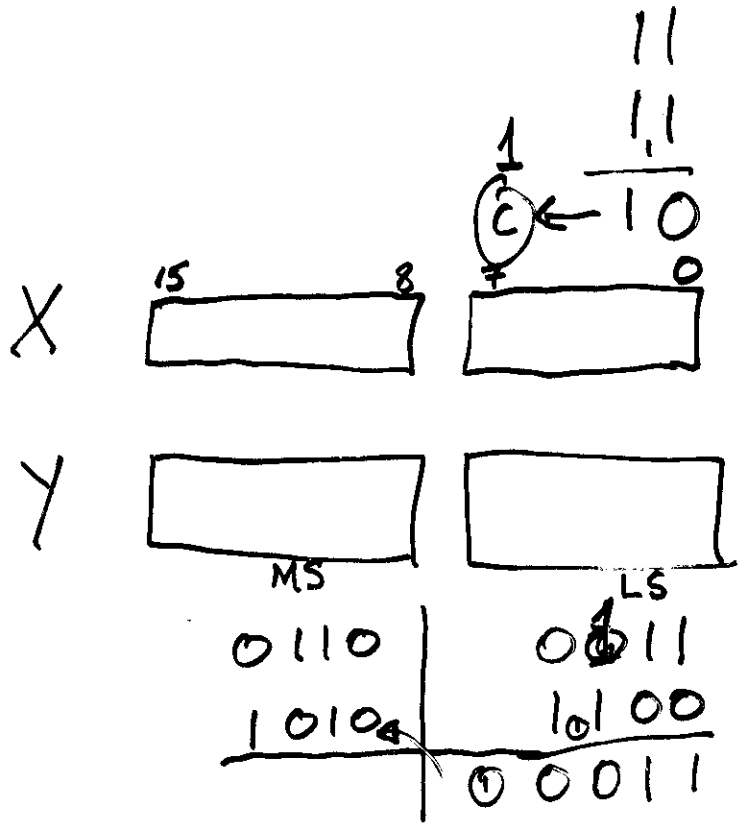
$$V = \overline{A_3}\,\overline{B_3}\,R_3 + A_3\,B_3\,\overline{R_3}$$

where A_3 = sign bit of 1st operand.
B_3 = sign bit of 2nd operand.
R_3 = sign bit of the result.

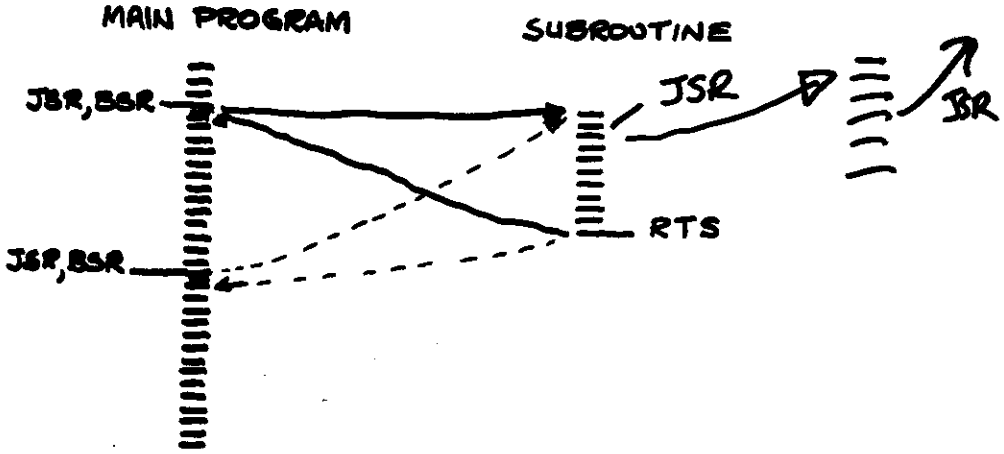
1ST EXERCISE (NUMBER 3.1)

- ADD TOGETHER 2 16-bit 2's COMPLEMENT NUMBERS TO PRODUCE A 16-bit RESULT
- TEST FOR OVERFLOW IN THE ADDITION PROCESS

  (V=1 AFTER THE ADDITION MEANS THAT THERE HAS BEEN AN OVERFLOW)



SUBROUTINES



- PURPOSE : TO AVOID REPLICATION OF SECTIONS OF CODE WHICH NEED TO BE REPEATED MANY TIMES IN A PROGRAM

- G800 PROVIDES SUBROUTINE ACCESS INSTRUCTIONS

'CALL' SUBROUTINE — JSR, BSR

'RETURN' — RTS

JSR : JUMP TO SUBROUTINE

INDEXED

MAIN PROGRAM PC		STACK SP		SUBROUTINE PC
n	JSR (AD)	→ SP-2		IX+K 1st INSTR.
n+1	OFFSET (K)	SP-1	(n+2) H	
n+2	NEXT INSTR.	SP	(n+2) L	

EXTENDED

PC		SP		PC
n	JSR (BD)	→ SP-2		AD 1st.
n+1	AD H			INSTR
n+2	AD L	SP-1	(n+3) H	
n+3	NEXT INSTR	SP	(n+3) L	

{ AD = Absolute Address }

BSR : BRANCH TO SUBROUTINE

PC		SP		PC
n	BSR (BD)	→ SP-2		n+2±K 1st
n+1	±K	SP-1	(n+2) H	INSTR.
n+2	NEXT INSTR	SP	(n+2) L	

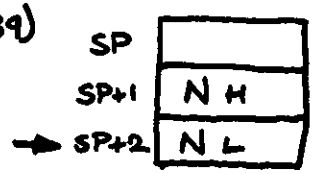
NOTE : THE STACKPOINTER MUST POINT TO A R/W MEMORY ADDRESS.

RTS : RETURN FROM SUBROUTINE

(7)

PC SUBROUTINE SP STACK

S RTS(39)



PC MAIN PROGRAM

N Next Instruction in the main program

SUMMARY

- RETURN ADDRESSES ARE SAVED ON THE 'PROGRAM STACK'
- THE SP MUST POINT TO A R/W MEMORY ADDRESS
- ENSURE THAT THE STACK DOES NOT OVERWRITE YOUR PROGRAM
- RESERVE ENOUGH MEMORY FOR YOUR STACK

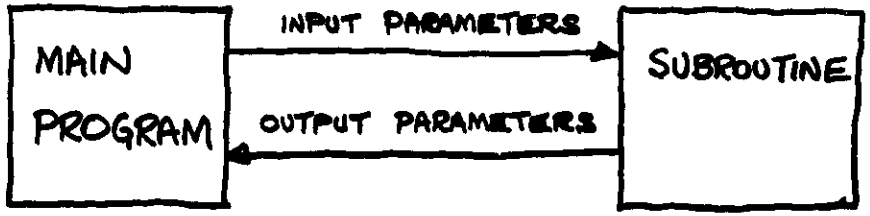
NOTE : THE MONITOR PROGRAM AUTOMATICALLY SETS THE SP TO A 'SENSIBLE' VALUE IN RWM.

24 byts

(12 addresses)

PARAMETER PASSING TO SUBROUTINES

(8)



- TYPES OF SUBROUTINE
  - NO PARAMETERS (FIXED DELAY)
  - INPUT PARAMETERS ONLY (VARIABLE DELAY)
  - OUTPUT PARAMETERS ONLY (RANDOM NUMBER GENERATOR)
  - INPUT + OUTPUT PARAMETERS (MULTIPLIER A = B*C)
- WANT MECHANISMS TO PASS PARAMETERS TO AND FROM SUBROUTINES
  - USE REGISTERS (A, B, IX)
  - USE POINTERS
    - IF MANY PARAMETERS ARE PASSED THEN USE MEMORY LOCATIONS TO HOLD THE PARAMETERS AND USE THE INDEX REGISTER TO POINT TO THE START OF THE PARAMETER BLOCK

MONITOR SUBROUTINES

- THE 6800 MICROPROCESSOR KIT CONTAINS A NUMBER OF SUBROUTINES WHICH MAY BE USED BY THE PROGRAMMER
- THE SUBROUTINES ARE CONTAINED IN ROM
- DETAILED DESCRIPTIONS OF ALL THE MONITOR SUBROUTINES ARE CONTAINED IN THE KIT USER MANUAL
  - SEE PAGES 4-1 TO 4-14
- PROGRAM LISTINGS FOR EACH OF THE SUBROUTINES ARE ALSO CONTAINED IN THE MANUAL
  - SEE PAGES 4-18 TO 4-54
- WE ARE ONLY GOING TO CONSIDER INPUT FROM THE KIT KEYBOARD AND OUTPUT TO THE KIT DISPLAY

KEYBOARD INPUT

RS	Fs	Fc	P/L	T/B
7	8	9	A	M
4	5	6	B	EX
1	2	3	C	RD
0	F	E	D	GO

KEYCODES

- 0-F 00-0F
- M 80
- EX 81
- RD 82
- GO 83
- Fs 84
- Fc 85
- P/L 86
- T/B 87

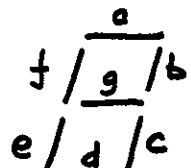
- ROUTINE CALLED GET WHICH ENTERS THE KEYCODE OF A 'PRESSED KEY' INTO A LOCATION CALLED KEY WHENEVER A KEY IS PRESSED. ALSO A LOCATION CALLED KYFLG ($E14C) IS SET TO 01 TO INDICATE THAT A KEY HAS BEEN PRESSED
- SECOND ROUTINE CALLED RDKEY ($FIEF) WHICH PERFORMS THE FOLLOWING FUNCTIONS
  - SETS KYFLG TO 00
  - RETURNS THE KEYCODE IN THE A-REGISTER TO THE MAIN PROGRAM

- THE ROUTINE GET USES THE NMI (NON-MASKABLE INTERRUPT) OF THE 6800
- INVISIBLE TO YOUR PROGRAM
- TO ENTER KEYCODES INTO YOUR PROGRAM
  1. TEST IF KYFLG IS EQUAL TO 01
  2. IF IT IS THEN CALL SUBROOTINE RDKEY. THE CODE OF THE 'PRESSED' KEY WILL BE IN THE A-REGISTER ON RETURNING FROM THE SUBROUTINE.

NOTE: YOUR PROGRAM SHOULD TEST KYFLG AT 'REGULAR INTERVALS' TO SEE IF A KEY HAS BEEN PRESSED.

DISPLAY OUTPUT

- THERE ARE 6 SEVEN-SEGMENT DISPLAYS ON THE KIT



- DISPLAY CODE

X	g	f	e	d	c	b	a
msb							lsb

- A '1' IN A BIT LOCATION CORRESPONDS TO A LIGHTED DISPLAY SEGMENT

- DISPLAY CODE

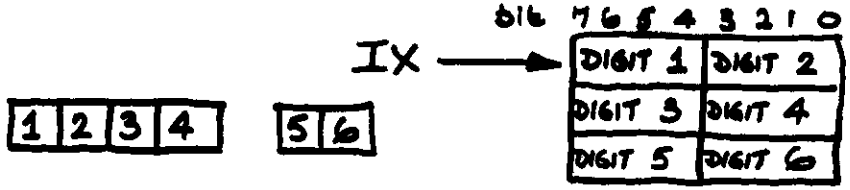
X	1	1	1	0	0	0	1

- MONITOR ROUTINE CLRDS ($F195)
  - CLEARS THE DISPLAYS ACCORDING TO A MASK IN THE A-REGISTER
  - A '1' IN A BIT POSITION 'BLANKS' THE DISPLAY

TO BLANK ALL SIX DISPLAYS

```
LDAA #%00111011
JSR CLRDS
```

- THE MONITOR PROVIDES ROUTINES FOR OUTPUTTING HEX DIGITS TO THE DISPLAYS
- TOO COMPLEX, SO WE HAVE WRITTEN OUR OWN
- SUBROUTINE CALLED DISPLAY ($E800)
  - DISPLAYS HEX DIGITS ON THE SIX DISPLAYS FOR A VARIABLE TIME
  - INPUT PARAMETERS
    - THE INDEX REGISTER POINTS TO THE START OF A TABLE CONTAINING THE HEX DIGITS TO BE DISPLAYED



  - YOU MUST DECIDE WHERE THE DISPLAY TABLE IS AND TELL THE 'DISPLAY' SUBROUTINE
  - THE B-REGISTER CONTAINS A NUMBER (0-255) WHICH SAYS HOW LONG THE DISPLAY IS ON FOR IN INCREMENTS OF 10mS.
    - IF B = 3 THE DISPLAY IS ON FOR 30mS AND THEN TURNED OFF.

NOTE : THE SUBROUTINE DOES NOT ALTER THE CONTENTS OF ANY REGISTERS

- TWO FURTHER EXERCISES

3.2 ADD 10 8-bit 2's COMPLEMENT NUMBERS AND DISPLAY THE RESULT.

3.3 INPUT 2 HEX DIGITS FROM THE KEYBOARD, ADD THEM, AND DISPLAY THE RESULT.