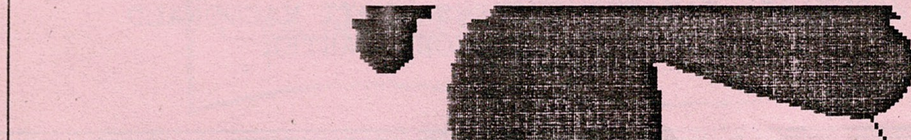
SEGAMAG
NEW SOUTH WALES

MARCH 1989



Registered by Australia Post
Publication No. NBG8403 x

$4.00

OFFICERS

President	Scott MacDonald	(046)668-956
Vice President	Rex Chandler	(02)872-4256
Treasurer	Brian Minett	(02)825-3998
Asst. Treasurer	Ted Hartley	(02)622-2416
Secretary	Allan Rodd	(02)816-1618

HELP DESK: Scott MacDonald(all hours)
John Carter(750-3271) 7-9pm

MEETINGS: "2ND SUNDAY OF EACH MONTH
NEXT MEETINGS: March 12; April 9; May 14

SERVICING: SCOTT MAC DONALD

HARDWARE: TAWARRI SYSTEMS=SCOTT MACDONALD
COMPUTRONICS=JOHN CARTER

LETTER TO THE EDITOR

Looks like everything is going very well since the downfall earlier in the year.

The meetings are getting much better each time with more varied and interesting activities. I like the idea of a timetable being set out for the day which shows good organisation brought by our staff.

As for the magazine, I can only say that it is getting better each time.

May I suggest that you put a letter to the editor in the beginning of the magazine because when ever I received the magazine I always looked forward to reading that section first.

Is there any chance that you could help me with a game called "Ninga Princess" as I cannot find all the scrolls which are needed to enter the temple. Your help would kindly be appreciated.

Well, that concluded what Ii have to say so keep up the good work!!

Fady Sarkis

EDITORS NOTE: Your request has been passed along to our games expert for a reply next issue.

AGENDA FOR MARCH MEETING

SUNDAY, MARCH 12, 1989, 10 am to 4 pm,
GLADESVILLE PUBLIC SCHOOL OPP. P.O.

10-11:30	Games, possible comp.
11:30-12:00	Discussion on future computor trends with Scott
12-1	Lunch
1-1:30	General discussion on future direction of group
4:00	End of meeting

COMPLIMENTARY TEA, COFFEE OR CORDIAL AT MEETINGS

CLUB TO EXPAND

The committee decided at a recent meeting to advertise for owners of other brands of PC's to join our users group. This move was a result of a poor response to renew memberships by Sega owners. To date we have only 40 paid members.

The committee also recognised the fact that many members also have a second computer of a differerent brand and would be interested in programs for both machines.

If you have friends with another brand of computer why not invite them to attend our meeting--free as a visitor--so we can get an idea of the interest in adding other brands to our membership. Many other brands operate on the MSX system similar to Sega so thare may be some immediate exchanges possible.

Advertisements have been placed in the Computors sections of the Sydney Morning Herald(March 4) and The Trading Post.

REVIEW OF FEBRUARY'S MEETING.

Another excellent discussion from Ted Hartley took place. This time, the topic was Modems. We were treated to a thorough introduction to the use of Modems for logging into Bulletin Boards to send and recieve messages and to download programmes. We were then taken through the uses of Viatel for looking up such vital information as a report on 'Minerals Futures'. Baud Rates and ASCII transfer were also explained. Reference was made to 'The New Hacker's Handbook' by Hugo Cornwall (ISBN 0-7126-9711-X) which gives comprehensive information on the use of modems from first principles. Anyone wishing further information on modems can write to THE EDITOR and Ted or another knowledgable person will reply. The discussion on Basic Programming was postponed through lack of attendance on the day. Anyone needing help with a particular programming concept can always receive assistance by writing to THE EDITOR or by seeing myself or another of the office bearers at the meetings. The rest of the day was devoted to informal discussions and games playing. The Master System again featured with some new games getting a good run. THUNDER BLADE, which features superb arcade style graphics, is an aerial combat action game using helicopters, and is recommended. We look forward to another interesting and successful meeting in March.

REX CHANDLER.

VICE-PRESIDENT.

CONTRIBUTIONS: Members contributions of letters, stories and programs are welcomed. Please submit printed copy ready for publication (SEGAWORD settings: 55-5-54-66). We can reduce or enlarge copy or graphics--new cover graphics needed for the next issue.

FINAL VERSION OF NOUGHTS & CROSSES
(See next page for dissection)

```
10 REM                        'NOUGHTS & CROSSES' TUTORIAL.
20 CLS:FOR I=0 TO 1:FOR J=0 TO 16:REM PRINT GRID & NUMBERS.
30 CURSOR 9+J,4+5*I:PRINT "/"
40 NEXT J,I
50 FOR I=0 TO 1:FOR J=0 TO 14
60 CURSOR 14+6*I,J:PRINT "/"
70 NEXT J,I
80 FOR I=0 TO 2:FOR J=0 TO 2
90 CURSOR 8+6*J,5*I:PRINT J+3*I+1
100 NEXT J,I
110 CURSOR  4,20:INPUT"";A$:REM ACCEPT A GO & CHECK LEGALITY.
120 A=VAL(A$)
130 IF LEN(A$)>1 OR A=0 THEN CURSOR 4,20:PRINT CHR$(21):GOTO 110
140 IF B(A)>0 THEN A=0:GOTO 130
150 FOR I=0 TO 2:FOR J=0 TO 2:REM WHERE TO PRINT O OR X.
160 IF A=3*I+J+1 THEN X=11+6*J:Y=2+5*I:B(A)=Z+1
170 NEXT J,I
180 IF Z=0 THEN B$="O":Z=1:GOTO 200:REM WHOSE GO WAS IT?
190 B$="X":Z=0
200 CURSOR X,Y:PRINTB$:REM PRINT THE GO ON THE GRID.
210 FOR I=1 TO 3:REM DETECT WINNER.
220 IF B(I)<>0 THEN IF B(I)=B(I+3) AND B(I)=B(I+6) THEN W=1
230 NEXT
240 FOR I=0 TO 2
250 IF B(3*I+1)<>0 THEN IF B(3*I+1)=B(3*I+2) AND B(3*I+1)=B(3*I+3) THEN W=1
260 NEXT
270 IF B(1)<>0 THEN IF B(1)=B(5) AND B(1)=B(9) THEN W=1
280 IF B(3)<>0 THEN IF B(3)=B(5) AND B(3)=B(7) THEN W=1
290 IF W=1 THEN CURSOR 4,20:PRINT B$;" WINS":W=0:GOSUB 430:GOTO 350
300 FOR I=1 TO 9:REM DETECT STALEMATE
310 IF B(I)=0 THEN S=1
320 NEXT
330 IF S=1 THEN S=0:GOTO 110
340 CURSOR 4,20:PRINT "STALEMATE"
350 CURSOR 4,18:PRINT "ANOTHER GAME? Y/N":REM SETUP NEXT GAME.
360 A$=INKEY$:IF A$<>"Y" AND A$<>"N" THEN 360
370 IF A$="N" THEN CLS:PRINT "BYE":END
380 FOR I=0 TO 2:FOR J=0 TO 2:REM BLANK LAST GAME'S GOES.
390 CURSOR 11+6*J,2+5*I:PRINT" "
400 NEXT J,I
410 ERASE
420 CURSOR 4,18:PRINT CHR$(21):A=0:Z=0:GOTO 130:REM CLEAR 'ANOTHER GAME'.
430 IF B$="O" THEN OW=OW+1:CURSOR12,22:PRINT "O =";OW:RETURN:REM PRINT SCORE.
440 XW=XW+1:CURSOR 21,22:PRINT "X =";XW:RETURN
```

SAVE TYPING--ALL PROGRAMS LISTED IN THIS ISSUE CAN BE COPIED ON YOUR DISCS OR CASSETTES FREE AT THE MEETING. BRING BLANK DISCS AND SEE REX CHANDLER AT THE MEETING

NOUGHTS & CROSSES TUTORIAL.

This is the dissection of the completed 'Noughts and Crosses' programme which has been developed over the last few meetings. The instruction REM has been used to give pointers to the function of the next few lines. All programmers should use this because its easy to get lost as programmes get larger or as time passes. The computer goes to the next line without taking action whenever it finds REM. The instruction : is used to put separate instruction on the same line. This is often done for tidyness or speed of execution, but it is also useful when a group of related instructions need to be executed after an IF-THEN test. In order to make the dissection easier to follow, I've simplified some of the formulae and altered the layout slightly from that previously published.

The follouing variables have been used:-

I and J	: Counters on the FOR-TO-NEXT loops.
A$	: Keyboard input.
A	: Value of keyboard input.
B(A)	: Array variable which stores the contents of the nine grid positions.
X and Y	: Grid co-ordinates.
Z	: Controls whose go and who owns each grid position.
B$	: Prints O or X on screen.
W	: Winner.
S	: Stalemate.
OW and XW	: O and X's score.

LINES 20-40: Two 'nested' FOR-TO-NEXT loops are used to print the horizontal lines of the grid. The formula uses the loop counters to calculate the cursor positions, ie (9,4),(10,4),(11,4) etc and (9,9),(10,9),(11,9) etc.

LINES 50-70: The process is repeated with a sligtly different formula for the vertical lines.

LINES 80-100: The formulae used in these loops count from 1 to 9, and calculates the position to print each grid number, again using the loop counters.

LINE 110: The INPUT command is used so that the programme waits for the player to type and enter the 'go'. The "" (null string) follows INPUT so that no message is displayed before the cursor. The entry is recorded in the variable A$. A string variable (which treats all entries as symbols with no numerical value, even if it represents a number) was chosen so that illegal entries could be detected more easily. If the variable 'A' had been used directly, entry of a letter would have caused the INPUT command to ask for a new entry and would have scrolled our carefolly prepared screen by one line.

LINES 120-130: The entry A$ is converted to the value A, and its legality is tested. Only values from 1 to 9 are permitted. Therefore the length of A$ must be 1. Similarly, if text was entered, the value of A is 0. Accordingly, if the IF-THEN test is true, the remainder of the line is executed and entries longer than one, and text entries are disallowed. The cursor is returned to the correct position and the control code 'PRINT CHR$(21)' is used to clear the line after the cursor. We then GOTO 110 in order to accept a new entry.

LINE 140: The array variable B(A) stores the values in each of the nine grid positions. The variable A can only be a number from 1 to 9 (see above). Therefore B(A) can only be B(1) through to B(9), depending on the last legal entry from the keyboard. The term 'Array Variable' comes to us from mathematics. When a programme is run, all variables are set at 0. Accordingly, B(A) will be 0 unless we have changed it, ie if no-one has entered a go in that square. If B(A) for that square was previously altered, we return for a new entry. The device used to achieve this goal is to let A=0 and GOTO 130 so that the IF-THEN test in that line is true.

LINES 150-170: The first formula in the loops counts from 1 to 9 and uses an IF-THEN test to see if the entry matches. When the test is true, the second formula calculates X and Y (the x and y co-ordinates of the square on the grid), and gives the square described by 'B(A)' a value of Z+1. Z is the variable used to determine whose go it is. If Z=0, it is O's go and if Z=1, it is X's go. On start-up, Z=0 and it is always O's go first. If B(A)=0, it means the square is not used yet. Accordingly, letting B(A)=Z+1 means that if B(3)=1, square 3 belongs to O, and if B(7)=2, square 7 belongs to X, etc.

LINES 180-200: If it was O's go, (Z=0), B$ becomes 'O', we let Z=1 so that the next go is X's, and GOTO 200. If the test was false, the reverse happens. Either way, a 'O' or a 'X' is printed at X,Y (the x and y co-ordinates determined earlier).

LINES 210-280: This section checks for winners and lets W=1 if three in a row values of B(A) are the same and not zero. Lines 210-230 check horizontally, 240-260 check vertically, and 270-280 check the diagonals.

LINE 290: If there was a winner (W=1) the current B$ which relates to the last go, is printed followed by the word "WINS" and W is reset to 0 in case there is another game. The GOSUB to line 430 keeps score. If O was the winner, the variable OW is increased by one, and likewise the variable XW for X. The winner and score are printed at the selected screen positions and the programme RETURNs to the command following the GOSUB (in this case GOTO 350 which sets up another game).

LINES 300-330: This section detects a stalemate by looking for unused squares and letting the variable S=1 if at least one is found. If one was found, the variable S is reset to 0, and the programme goes back for a new INPUT. If not, the word STALEMATE is printed and the programme continues onto Line 350. ▪

LINES 350-370: This section asks whether a new game is to be set up, using the INKEY$ command. This differs from the INPUT command used earlier because the programme only checks the keyboard once before moving on. To overcome this, Line 360 is a loop which will only permit an exit if the "Y" or "N" buttons are pushed. The button pushed is not displayed on the screen, and there is no need to press CR or do error checking for incorrect keys. If "N" is pushed, screen is cleared and the programme is broken using the command END. If "Y" is pushed, the programme continues on to Line 380.

LINES 380-410: The O's and X's on the grid from the previous game are removed by printing a space over each one. This is done using the nested loop. The values in the array variable B(A) which controls who owns each square, are all set to zero using the command ERASE. This command sets the value of all array variables to zero.

LINE 420: This line removes the message "ANOTHER GAME" using the device described earlier and lets Z=0 so that 0 will have first turn. It lets A=0 and goes to Line 130. Because A=0, the test at line 130 is true and this removes the message telling who won or stalemate. The programme is then ready for the first INPUT of the next game.

I hope I've been able to introduce these programming concepts in a useful way so that you can now apply them to your own endeavours. I'll be available at the meetings to assist further, and can also be contacted via a letter to the Editor.

Happy Programming,
REX CHANDLER.
VICE-PRESIDENT.

CLUB LIBRARY

Yes we are planning to (eventually) get our club library into action. The library will have the items below available for hiring at special club rates:SF-7000 disc drives, SC3000 keyboards, SM1200 modem and software.

Hiring charges are yet to be confirmed but will be determined at our next committee meeting and published in the next newsletter. Another good reason to be a financial member of the Sydney Sega Users Group.

John Carter, Librarian

SEGA COMPUTER GAMES REVIEW

This month we will review Castle which is available on cartridge by Sega.

Castle is an arcade quality adventure game similar to Sir Rodericks Quest style of game where you controll a little man through many screens (rooms), but castle does not lack speed, in fact there are two speed settings and the faster speed is really fast for an adventure game!

There is 100 different rooms to explore to finish the game but to keep advancing to each room you will have to collect keys to open doors, collect and use oxygen tanks for swimming under water, ride elevators, jump deformed creatures, ride conveyor belts, collect a map, climb boxes, move barrels, controll cargo lifts, ride moving platforms, collect gold! and visit the invincible power plant for short invincibility to name many but not all of the different tasks involved to make it through this game. The animation and graphics are great with a nice tune playing in the background.

This game is a great game but by no means a push over, you need to be carefull not to waste the keys that you collect or you will not get through all of the rooms and for those of you who have already bought the game, there is a continue game feature- when you lose your last life, press both HOME CLR and INS DEL and hold down untill the game over symbol changes to continue, this will allow you to continue with the keys you had and a full set of lives. However I still do not know of anybody who can get through this game and there are quite a few owners of this game, so if anybody has made it through this game then pleasewrite into our SEGAMAG and give us all a few clues PLEASE!!!

For those of you who would like to purchase this excellent game then it may be purchased from-

Computatronics- P.O.BOX 17 Ryde 2112 $50.00 + $2.50 postage/packaging and handling.

Tawarri Systems (Scott) at the club meeting for $50.00

Reviewed by,
John Carter.

```
1 REM PELLET MAN by Jonathan Kirkwood 1987
2 SC=0:RD=1:CLS:COLOR1,15:CONSOLE0,24:CLS
3 PT=0:YX=15:YY=15:X1=5:Y1=5:X2=30:Y2=19
4 CURSOR0,0:PRINT"SCORE :";SC;"       HI-SCORE :";HS
5 CURSOR0,21:PRINT"Jonathan Kirkwood 1987":CURSOR0,20:PRINT"PELLET MAN"
6 FOR I=1 TO RD*20:X=INT(RND(1)*36):Y=INT(RND(1)*19)+1
7 ZZ=VPEEK(Y*40+X+2+&H3C00):IFZZ<>32THEN 6
8 CURSORX,Y:PRINT".":NEXT
9 CURSORYX,YY:PRINTCHR$(253):CURSORX1,Y1:PRINTCHR$(250):CURSORX2,Y2:PRINTCHR$(250)
10 A$=INKEY$:IFA$=""THEN27
11 BEEP1:BEEP0:CX=YX:CY=YY
12 IFASC(A$)<28ORASC(A$)>31THEN27
13 IFA$=CHR$(28)ANDYX<35THENYX=YX+1
14 IFA$=CHR$(29)ANDYX>0THENYX=YX-1
15 IFA$=CHR$(30)ANDYY>1THENYY=YY-1
16 IFA$=CHR$(31)ANDYY<19THENYY=YY+1
17 YV=VPEEK(YY*40+YX+2+&H3C00)
18 IFYV=250THEN44
19 IFYV=32THENCURSORCX,CY:PRINT" ":GOTO 27
20 CURSORCX,CY:PRINT" ":CURSORYX,YY:PRINT CHR$(253)
21 IFYV<>46THEN27
22 BEEP:SC=SC+100+INT(RND(1)*51):CURSOR0,0:PRINTCHR$(5)
23 PT=PT+1
24 CURSOR0,0:PRINT"SCORE :";SC;"       HI-SCORE :";HS
25 IFPT=RD*20THENCLS:BEEP:BEEP:BEEP:GOTO3
26 GOTO27
27 X3=X1:Y3=Y1:X4=X2:Y4=Y2
28 IFYX<X3THENX3=X3-1
29 IFYX>X3THENX3=X3+1
30 IFYY<Y3THENY3=Y3-1
31 IFYY>Y3THENY3=Y3+1
32 IFYX<X4THENX4=X4-1
33 IFYX>X4THENX4=X4+1
34 IFYY<Y4THENY4=Y4-1
35 IFYY>Y4THENY4=Y4+1
36 V1=VPEEK(Y3*40+X3+2+&H3C00)
37 V2=VPEEK(Y4*40+X4+2+&H3C00)
38 IFX1=YXANDY1=YYTHEN44
39 IFX2=YXANDY2=YYTHEN44
40 IFV1=46ORV2=46THEN43
41 IFV1<>46THENCURSORX1,Y1:PRINT" ":X1=X3:Y1=Y3
42 IFV2<>46THENCURSORX2,Y2:PRINT" ":X2=Y4:Y2=Y4
43 GOTO9
44 CLS:CURSOR13,10:PRINT"GAME OVER":FORV=200TO110STEP-10:SOUND1,V,15:SOUND2,V+1,15
:NEXT:BEEP:FOR A=0 TO 500:NEXT
45 IFSC>HSTHENHS=SC
46 GOTO2
```

PUBLIC DOMAIN PROGRAMS ARE AVAILABLE FREE TO MEMBERS AT THE MEETINGS. BRING YOUR BLANK DISCS AND CASSETTES AND SEE JOHN CARTER OR REX CHANDLER

MACHINE CODE-GETTING STARTED.

Programming in Machine Code gives the advantage of great speed of execution compared with Basic. Accordingly, much more can be made to happen on the screen and intricate programmes can be developed. In Machine Code, you have to be meticulous to give instructions in very fine detail and to look after the housekeeping, both of which are often built into Basic commands. I will present a personal view of Machine Code as a beginner myself, and will try to unravel some of the mysteries to help in getting started.

Machine Code instructions are always in hexadecimal, usually one or two bytes long, but sometimes three or even four. Some hexadecimal bytes used are also data which go with particular commands, for example, the coded instruction 3E 5F means LD(A) &H5F, or load the accumulator with the hexadecimal number 5F (ie &H5F). Once these coded instructions have been decided upon, they must be entered into the computer's memory so that they can be CALLed upon to execute when needed. The Machine Code storage area starts at the address &HF000. There are two ways of going about this. A programme in Basic can be made up which POKEs the commands into the memory locations from &HF000 onwards and then runs the programme by CALLing the relevant memory location. Examples are 'ARCADE' in SEGAMAG, SEPT 1988, and 'LIFE' in NOV 1988. An easier way is to buy an ASSEMBLER programme which lets you enter the symbols for the command (called mnemonics - yes thats spelt correctly). It then looks up the correct hexadecimal code for you and will also load your final programme. This second method is called ASSEMBLY LANGUAGE. A recommended one such as IKELA from TAWARRI will also check for technical errors and will do some housekeeping. Once you've loaded your programme and before you run it, you need to check that every detail is correct. To do this, a dissassembler is used. A public domain dissassembler was printed in SEGAMAG, JULY 1987. IKELA also has a built-in dissassembler function. This gives three columns of information such as:

```
F000 3E5F     LD A &H5F
```
(sometimes # is used instead of &H).

(a) F000- the memory location where this instruction starts,

(b) 3E5F- the hexadecimal instruction and data associated with it, and (c) LD A &H5F- the symbolic description (or mnemonic). You can then run your programme by CALLing whichever memory location is its starting point, but keep your finger near the on/off swich because some nasty 'crashes' can occur with faulty programmes. Remember to save your Basic programme first just in case.

The mathematical language you and I understand is DECIMAL, and is based on the number ten. The language of the computer is BINARY and its base is one. It stores and uses information as a series of zeros and ones (ie ons and offs) in different combinations. Each data point is called a bit (binary digit) and eight bits are grouped together as one byte. Because no-one can be expected to understand a long series of zeros and ones, a code was devised for each byte and this is called Machine Code. Two digits are used to define each byte, the right being for the right four (least significant) bits and the left for the left four (most significant) bits. If, moveing from right to left, we give four bits values of 1, 2, 4, and 8, we can see that combinations of zeros and ones can be added together to give any number from zero to fifteen. Because of this grouping of sixteen, the language of Machine code is called HEXADECIMAL. In order to keep one symbol for numbers above nine, the numbers 10-15 are represented by the letters A-F. When two hexadecimal digits are brought together to form one byte, 266 (ie 16*16) numbers (zero to 255) can be described. Have you ever wondered why Screen 2 is 266 pixels (bits) wide?

Another concept to understand before we tackle some of the more widely used commands, is the Basic Interpreter. This is LEVEL 3 BASIC, for cartridge users and DISC BASIC for disc drives. Both reside in memory from &H0000 onwards and contain all of the Machine Code routines for identifying and processing commands and other inputs in the BASIC language and outputting results. These routines can be accessed directly using the command CALL if you know the correct address. As describing these is a substantial undertaking, I shall save this topic for later.

The symbolic architecture of the computer and particularly of the Central Processing Unit (CPU) must be understood if programming in Machine Code is to be successful. The computer consists of a number of chips such as the CPU, the VideoRAM Chip, the Sound Chip, and the Communication Ports (RS-232 and Centronics). The CPU consists of memory locations from &H0000-&HFFFF. It is here that the Basic Interpretor and Basic and Machine Code programmes are stored and executed. The VideoRAM chip also consists of memory locations which can be accessed. It is here that information which describes the current screen display is stored. Changeing the information on this chip (such as by using VPOKE) will alter the display. If you use POKE, information on the CPU will be changed. In Machine Code, OUT commands and/or CALL routines are used to update the VideoRAM chip from the CPU, (and also the Sound chip and the Ports). IN is used for transfer of information into the CPU.

The main activity of the CPU centres on the Registers. These are given symbolic names as follows:

AF	Accumulator-Flags,
BC	Usually used in counting,
DE	General purpose,
HL	Usually used to specify addresses.

Other registers which you will not use very often are:

SP	Stack Pointer,
PC	Program Counter,
IX	X index register,
IY	Y index register,
I	Interrupt vector register.

We will deal with the first four only. Each can be used singly or in pairs except AF which always goes together. A single register contains one byte and a register pair contains two bytes. The Accumulator is where mathematical calculations take place. The Flags register records the status of the outcome of these calculations. Since the Flag Register cannot be used directly for calculations, AF can deal with one byte only. The Zero Flag, the Carry Flag and the Sign Flag are the most important ones and by examining the Flags Register, we can determine whether the result was zero, required a borrow/gave a carry, or was plus/minus. The accumulator deals with 8-bit arithmetic. There are commands which can be used for 16-bit arithmetic which use HL as an 'Accumulator' and record the status of the outcome in the Flags register. The Stack is an area of memory which is set aside for the storage ot data for later usage. The contents of any register can be PUSHed onto the Stack and later POPed back into the same or another register. The system is last-on-first-off so if other data has been added to the stack since the required number, this other data has to be removed before the original number can be recovered. Memory locations selected by the programmer can also be used to store data. The commands EXX and EX AF,AF' can also be used. AF' is the auxilliary register to AF and is used in the same way but not at the same time. Simliarly, the auxilliary registers BC', DE' and HL' are accessed using EXX.

It remains now to examine some examples of the more useful commands used for the loading, interchange, manipulation and output of data. These are described much more fully in the series called 'Understanding Assembler' Parts 1-5 by Scott MacDonald in SEGAMAGS June 1987, July 1987, Sept/Oct 1987, Nov/Dec 1987, and May 1988. I propose only to give a lead into these excellently detailed references, not to duplicate them. Another useful reference is 'Z80 Assembly Language For Students' by Roger Hutty. It can be obtained inexpensively from Dymocks (ISBN 0-333-32295-9) and it gives a good, easy reading, elementary description of Machine Code instructions, and binary and hexadecimal, with exercises. The definitive text on Machine Code is 'Programming the Z80' by Rodnay Zaks (ISBN 0-89558-06905). It is a thick, expensive and hard to follow tome, and is best used as an occasional reference.

In general, it is much more useful to determine exactly what it is you need to do in Machine Code and then look up your available commands to see which one(s) will do the job, than it is to try to learn all of the commands and then hope for insight on how they go together. There is always a command or a group of commands which will give the desired result and you are limited only by your imagination.

The Loading of registers is represented by LD and brackets are used to denote 'the CONTENTS of the ADDRESS indicated'. Thus :-
LD B,A is Load the B register with the contents of A. In this action, the content of A remains unchanged. Its value is copied into B.
LD A,(HL) is Load the accumulator with the contents of the memory location pointed to by HL. Find the current value of HL to see which memory location this is.
LD (F2E8),DE Loads the contents of E into the memory location F2E8, and of D into F2E9. The 'least significant' byte goes to the first memory location.
LD BC,&H5EF4 is Load B with &H5F and C with &HF4.
Remember, if you're not using an assembler, you need to reverse the bytes of numbers and of absolute memory locations. Thus :-
The hex for LD (F2E8),DE is ED 53 E8 F2
and for LD BC,&H5EF4 is 01 F4 5E.
There are numerous variations on the LD command other than these examples.

Jumping to another memory location is achieved using JP (Jump Absolute) and JR (Jump Relative). JP is the Machine Code GOTO. Thus JP &HFE4C means jump to memory location FE4C. This can be conditional on the Flags (zero, minus, carry and parity). Accordingly, JP NZ,&HFE4C means jump to FE4C if the zero flag is not set. JP(HL) also works. This means jump to the address pointed to by HL. JR e means jump forwards or backwards by the number of addresses specified by e. The jump is forward if e is between 0 and 7F and backwards if e is between 80 and FF. Conditional jumps on the zero and carry flags only can also be made. Thus JR C,FB is Jump -5 if the Carry flag is set. Beware of JR FE as this is an infinite loop.

DJNZ e, is a very useful command for determining the number of times a loop is repeated. It stands for Decrement B and Jump Relative by e memory locations if non-zero.

CALL is the equivalent of GOSUB and, like GOSUB, it is always followed by RET. As the return address is PUSHed onto the stack by the CALL command, it is vitally important that the sub-routine does not alter the Stack or, if so, returns it to its original condition. If not, the programme gets horribly lost (I speak from experience). CALL and RET can both be used conditionally. Thus CALL M, FE4C means GOSUB to FE4C if the last calculation gave a negative result. The Hex is FC 4C FE (with the address bytes swapped around if you're not using an assembler). RET NC means return if no carry resulted from the last calculation.

ADD A,B adds the value in B to the value in A and leaves the result in A with the Flags set. ADD HL,BC performs the same action using HL as the 'Accumulator'. Numerical constants or the value of an address pointed to by (HL) can also be used. ADC is a very similar command except that the Carry flag is also added to the result. SUB A,C subtracts C from A and leaves the result in A. There is no equivalent command for SUB HL. SBC A,B subtracts B from A then subtracts the carry flag. This time the equivalent SBC HL exists.

CP B will compare the value in B with that in A without altering either value but with the Flags affected. Thus the CP command is usually followed by a conditional Jump or Call.

BIT 5,B tells us whether bit 5 of the B register is zero or one, by altering the Zero flag without altering any of the registers. It also is usually followed by a command conditional on zero. All register Bits can be checked in this way.

Further commands deal with rotate, shift, logical operations, block shift, block search, input and output, setting and resetting bits, conversion to decimal arithmetic, etc. My aim was to provide an introduction to the structure of Machine Code. Accordingly, the remaining commands are left to other references. The next step might be to puzzle out how these commands can be put together to perform simple functions and to analyze simpler programmes. I'll be happy to answer any enquiries as best I can either written to me c/- THE EDITOR, or at the club meetings.

Until then,
HAPPY HEXHUNTING,
REX CHANDLER (VICE PRESIDENT).

CONTRIBUTIONS: Members contributions of letters, stories and programs are welcomed. Please submit printed copy ready for publication (SEGAWORD settings: 55-5-54-66). We can reduce or enlarge copy or graphics--new cover graphics needed for the next issue.

Postage
Paid
Australia

SEGAMAG NEW SOUTH WALES
PUBLISHED BY SYDNEY USERS GROUP
PO Box 455
GLADESVILLE N.S.W. 2111

20
60

MR ALLAN RODD
19 CHAMPION RD
GLADESVILLE
2111 N.S.W.