AUNTIE JOHN

Looping the loop

Time to start pushing and jumping in Aunty John's action-packed machine code series.

Last month, we finished off by looking at an example of decision making in machine code. In Basic, this program might look like:

```
10 IF PEEK (&90000) = 10 THEN GOTO 100
20 IF PEEK (&9000) = 11 THEN GOTO 150
30 POKE 9001,0
40 STOP
100 POKE &9001,42
110 STOP
150 POKE &9001,67
160 STOP
```

Not a very exciting program, I agree, but it proves a point.

Slightly – but not much – more interesting is the prospect of allowing the program to loop back on itself. Such a program is shown below:

```
LDA 1,10
LOOP: NOP
DEC A
CPO
JR Z.LOOP
JR STOP
```

It's short, succinct and useless. At least it introduces a new instruction to us: NOP, or No-OPeration. Guess what it DOES? That's right – nothing. Believe it or not, NOP (code &00 – yes, zero) can sometimes be useful, so I'm reliably informed.

The program sets A up to be a counter, and then subtracts one from it repeatedly until it is equal to zero. Our first loop.

This is our first dealing with any proper programs – programs that make decisions and go places. The sort of program that would do well in business. The sort of program that you

wouldn't like to introduce your girlfriend to in case she fancied it more than you.

The machine code numbers for jumps are all very interesting, but if you use them you'll need to work out the relative jump distances yourself. This is a soul-destroying task and there is no reason why anyone should do it these days – assemblers are easy to come across.

Now for something good, something that provides scope for lots of silly mental images and almost explains a great fundamental of machine code – how to stop it from crashing at the end. Viewers in Northern Ireland will have

their own programme.

Here is a problem for you. What do you do when you run out of registers? Say you are using the registers in their paired up, two byte, 16 bit form. And say that you need to store another number, but you have no more registers to store it in. Bit of a bummer, eh? You could take each single byte register in turn and store it in a memory location, but that seems too much effort. There must be a better way.

Indeed there is a better way – it's called a stack. It is such a fundamental part of computer programming that entire books have been written about stacks (I know, I had to read them at university – they are the most boring books ever published). So what are stacks? And how are they attached to the back of the computer? Do they connect to the cassette interface? What colour are they? Do they have flashy lights on them?

Normally stacks are explained using analogies with paper plates or playing cards. I don't know why; they just are. I prefer to explain it using Pink Floyd albums.

Take your collection of Pink Floyd albums, and carry them into the kitchen to the kitchen table. The kitchen table is going to be the computer memory, where we can store the albums. Now you feel that your arms are getting tired holding all these albums, so you put Dark Side of the Moon on the table. You have PUSHed the album onto the stack. Then you put Wish You Were Here on top of it. You have PUSHed another object onto the stack. Suddenly you feel a great desire to listen to Money which is on Dark Side of the Moon as you well know. So you lift Wish You Were Here – POP it off the stack – and then pick up Dark Side of the Moon – POP it as well.

That's what a stack is – an ordered pile. It's called a 'first in last out' stack because that is what happens – the first item put in is the last item to be taken out. The golden rule is that it is impossible to touch any object other than the one on the very top of the stack. You can put more objects on top of it but you can't touch any underneath it without moving the top one first.

So what does this have to do with machine code? Quite a lot actually. The instructions PUSH and POP will work with the register pairs to store and retrieve items from a stack. For example:

```
PUSH HL
;do something else
POP HL
```

will always ensure that the initial value of HL is preserved.

Here is a tricky problem. Look at it carefully, and see if you can guess what will happen. Look very carefully – there is a trick.

```
PUSH BC
PUSH DE
;do something
POP BC
POP DE
```

Looks ok does it? Preserves BC and DE does it? Wrong! It doesn't! It actually swaps the initial values of BC and DE. Think of the stack – first BC is put on it, then DE. After the 'do something' bit a value is POPed and put into BC; this value is the last value to have been placed on the stack, that of DE. Then another value is POPed and placed into DE – the value that was in BC.

The important thing to remember is that the registers used in the POP command only determine where the value on the top of the stack is to be moved to. Once the values are on the stack, where they came from is irrelevant. Be very careful with stacks and double check the order of PUSHes and POPs. You MUST also remove all the values you have PUSHed before your program or routine ends. This is vital – even if you don't think you need the values again, you absolutely must POP them off the stack. The next chapter has the reason why.

The more cynical of you might wonder what would happen if you were to POP a value of the stack before you had PUSHed one up. The answer is garbage. The system would probably not crash (immediately that is) but the value returned would be one which you didn't really expect. (This is a plug for the next chapter – get it or be forever in the dark). The even more cynical might wonder what would happen if you kept on PUSHing things onto the stack. Would the numbers pile up inside and wear out through cracks in the keyboard? Sorry to disappoint you. If you did stack over twenty thousand numbers then all the spare REM would be used up and the computer would crash.

Not really recommended.

To summarise, stacks are like piles of Pink Floyd albums – very desirable. The Z80 allows you to store the numbers on a stack, as long as you promise to take them off when you finish. There were hardly any listings and next to no jokes. Bit dull really.