Z80 Assembler – Part 1

Introduction

This is the start of a series of tutorials / challenges to help people get into the world of Z80 Assembler. I do not say what I show here are the best / fastest ways to do things, in fact often the opposite. The aim is to give easy to follow examples followed by a series of challenges and questions. Everyone can then either share their answers onto the Facebook page, or if they prefer email the myself at adrian@apbcomputerservices.co.uk – I will endeavour to answer any questions and comment on any posts. Remember often there is no single right answer, often any answer that gives the correct result is right. Never feel ashamed or worried to post to the Facebook page, we have a strict policy of being welcoming and supportive, those not following those rules will be dealt with, so post away, ask questions, it's the best way to learn.

I'm creating these for you! If nobody posts any answers, asks questions etc, then ill assume no one is interested, its up to you all. If you have any suggestions, then let me know. Ill upload a file containing the main examples in code and pre-assembled tap. Format. To assembler the code simply double click the build.bat file or load the .sln into visual studio and you should be able to use the build command. So, onto Part 1… printing text to the screen.

Getting Started

One command most people will remember from BASIC is CLS, this command will clear the screen. There is a ROM routine that does this that sits at memory location 3435 (0x0D6B).

```
CALL    3435
RET
```

If you assemble this and run it, you will find that it (as expected) clears the screen and returns us to the basic prompt.

We are now ready to print some text on the screen. This can be done by the useful RST 16 command which will print the character in the accumulator. With this in mind lets expand our example.

```
CALL    3435
LD      A, 'G'
RST     16
RET
```

Again, assembler and run it and….. Hang on!! I said RST 16 prints a character, this did nothing… Ok, technically it did, but the ROM routine 3435 leaves the output set to channel 1 (the lower part of the screen – See TECH INFO #1 for channel information).

For those that don't believe me, if we replace the RET with an infinite loop, so just replace it with

```
Loop:   JR      Loop
```

And re-assemble and run the program, you will see the 'G' is printed down at the bottom of the screen. While that is correct, its not exactly what we wanted. To output to a different channel we can use the ROM Routine at 5633 (0x1601) CHANN_OPEN to change the channel. In most cases we want to output to the screen which is channel #2, to do this we just need to add

```
LD      A, 2
CALL    5633
```

After our CLS call and before we do RST 16. Presto, our letter now appears at the top of the screen. We can actually achieve the same thing by using a slightly different CLS ROM routine called CL_ALL, this is at memory location 3503 (0x0DAF). This clears the screen but leaves the current channel set to channel #2. Using this we can remove the need for the call to 5633.

Printing strings

Now we have printed a single letter, we can use RST 16 do print extra characters. This is simply a case of loading A with the correct value and calling RST 16 again and again

```
        LD      A, 'H'
        RST     16
        LD      A, 'e'
        RST     16
        LD      A, 'l'
        RST     16
        LD      A, 'l'
        RST     16
        LD      A, 'o'
        RST     16
        LD      A, ' '
        RST     16
        LD      A, 'W'
        RST     16
        LD      A, 'o'
        RST     16
        LD      A, 'r'
        RST     16
        LD      A, 'l'
        RST     16
        LD      A, 'd'
        RST     16
```

Well, its long winded but it gets the job done. Surely there are better ways to do it.. Well of course there is, we can store the string in memory and use a loop to write the text.

```
        LD      HL, String
        LD      B, 11
Loop:   LD      A, (HL)
        INC     HL
        RST     16
        DJNZ    Loop
```

```
        RET
String: DB   'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', ' r', 'l', 'd'
```

That's a little better, and we could easily use a similar function over and over again to print other strings if we were to pass the pointer to the string we wanted printing.

Printing extra characters

So far we have only been printing standard Alpha-numeric characters, that is the letters A-Z and numbers 0-9. There are various other characters you can print using RST 16, below are some of the more common ones.

13 (0x0d) – New line, this is like pressing enter, it brings the cursor back to the start of the next line down.

16 (0x10) – INK n, sets the ink colour to the value next written to the stream.

17 (0x11) – PAPER n, like ink, but alters the paper colour.

18 (0x12) – FLASH n, sets flash on (1) or off (0).

22 (0x16) – AT x,y, changes the position of the cursor to the x,y coordinates which are the next two values written to the stream.

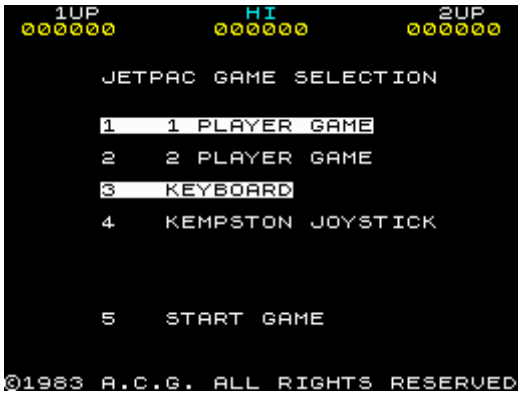
Using these commands we can start to play with the text more, we could expand our previous example to

```
        LD      HL, String
        LD      B, 14
Loop:   LD      A, (HL)
        INC     HL
        RST     16
        DJNZ    Loop
        RET
String: DB      22, 5, 5, 'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', ' r', 'l', 'd'
```

This would have the effect of moving the text to position 5, 5 on the screen. Note we had to change the value of B to ensure we drew all the characters.

Conclusion

This has been a relatively short tutorial on printing to the screen, but it gives you some power to use strings in your programs. There have been a few games that have used this functionality to produce main menus etc.



Hopefully with the information provided above you can see how this screen could start to be created.

As mentioned, any questions or comments please let me know, below are a few questions and challenges. Comment on the Facebook page or email me at adrian@apbcomputerservices.co.uk and ill aim to respond as soon as possible.

Challenges

1) Given the function that uses a loop to print the “Hello World” string, can you expand this into a function that you pass in HL as the address of the string and B as the length of the string. Use this to print several strings.
2) Add the ability to pass in the X and Y location of the screen in the D and E registers
3) Can you alter the function so you don’t have to pass in the length, think of different ways you can figure out if you have reached the end of the string.

Extra Credit

As an additional challenge, can you set up a menu draw function that you pass in a pointer to a menu. This will then draw a menu similar to that of the Jet Pac screen, it must have the 5 options

1 1 PLAYER GAME

2 2 PLAYER GAME

3 KEYBOARD

4 JOYSTICK

5 START GAME

Tech Info

Tech Info #1: There are 3 main channels that are used, they are

1 – (K Channel) Used to write to the bottom 2 lines of the screen, this is similar to the INKEY$ function in BASIC

2 – (S Channel) This is the main screen channel, we have been using this in the code above.

3 – (P Channel) This will write out to a printer if one is attached, the above code can easily be changed to actually print to a ZX Printer.