Department of Electrical Engineering, IIT Bombay

EE309 Computer Organization, Architecture and Microprocessors: Tutorial Sheet III

8085 Programming

1. **Quick Miscellany: 8085**
   - (a) What is the maximum number of input and output devices that can be connected to the 8085 ?
   - (b) Write two pieces of code to read in the contents of (i) the stack pointer (SP), and (ii) the Program Counter (PC).
   - (c) Write the smallest program segment (corresponding to the minimum number of bytes) to exchange the contents of two 16-bit registers - the BC pair and the DE pair.
   - (d) i. Enumerate at least 5 one-byte commands (apart from the `CMC` instruction) which cause the Carry (CY) Flag to be reset – Note that for this question, the same basic command applid to two differnet registers will not be treated as different commands.

     ii. If one is not permitted to use the `STC` command or any explicit subtraction command (`SUB, SUI, SBB, SBI`), what are the fastest ways to set the carry flag ? Enumerate all. 'Fastest' implies optimality in the total number of T-states; and out of two programs having the same number of T-states, one corresponding to a smaller number of bytes will be deemed to be the faster one. Note that for this question, two programs with just the data fields different will be counted as being the same.

2. **Question with Strings Attached** Implement the following variant of the *strcpy* function: `void strcpy(char * src, char * dest)` Assume all strings to be already allocated, and that the strings do not overlap.

3. **More strings** . . . Implement the following variant of the *strncpy* function: `void strncpy(char * src, char * dest, unsigned char n)` Assume all strings to be already allocated, and that the strings do not overlap. Use *src*, *dest* as 16-bit addresses, and n as a 8-bit number. Note that in *strncpy* not more than `n` bytes are copied. Thus, if there is no null (`0`) byte among the first `n` bytes of `src`, the result will not be null-terminated. Further, in the case where the length of `src` is less than `n`, the remainder of `dest` will be padded with nulls.

4. **The Highs and Lows** Consider the BC register pair. Suppose you've forgotten which one holds the most significant byte (*a highly dangerous thought, indeed*). Write a small routine to test this out. If B contains the most significant byte, write FFh into A, else write 00h into A. Explain your logic.

5. **Acting Smart** A small bank has a SmartCard system for all its employees. The locker room access is controlled by an 8085-based microcomputer system. Each bank employee has a 1-byte number on his or her SmartCard. Only `n` employees have permission to use the locker room (the door closes automatically after an employee goes in). `n` is specified in a byte at memory location 2000h. A list of `n` codes ($\mathtt{n} \geq 0$) is stored from location 2001h onwards. Write a program segment with an infinite loop that uses three subroutines - `read_card`, `operate_door` and `check_validity`. Assume that the first two are given to you. `read_card` interfaces with the SmartCard reader on the door lock, and returns the SmartCard number in register B. `operate_door` uses the number in A prior to the call. If A contains FFh, it unlocks the door, else if A contains 00h, the door stays locked. In addition to the program segment above, write a suitable routine for `check_validity`,

6. (*Gaonkar*, Chapter 6)

   (a) What is the output at `port_1` when the following instructions are executed ?

   ```
             MVI A, 8Fh
             ADI 72h
             JC display
             OUT port_1
             HLT
   display:  XRA A
             OUT port_1
             HLT
   ```

   (b) If the instruction `ADI 72`h is replaced by `SUI 67`h, what will be the effect ?

7. (*Gaonkar*, Chapter 6) Explain the function:

   ```
             MVI A, byte_1
             ORA A
             JM outprt
             OUT 01h
             HLT
   outprt:   CMA
             ADI 01h
             OUT 01h
             HLT
   ```

8. **Looping the Loop** How do you think *large* delays are implemented ? Obviously, having a large number of `NOP` statements is not a nice method, as the program size will become large (in terms of the number of bytes).

9. (*Gaonkar*, Chapter 7) The following instructions are intended to clear ten memory locations starting from the memory address 0009h. Explain why a large block of memory will be erased or cleared, and the program will stay in an infinite loop.

   ```
          LXI H, 0009h
   loop:  MVI M, 00H
          DCX H
          JNZ loop
          HLT
   ```

10. (*Gaonkar*, Chapter 7) Loop: how many times ?

```
       LXI B, 0007h
loop:  DCX B
       MOV A, B
       ORA C
       JNZ loop
```

11. (*Gaonkar*, Chapter 7) What is the mathematical operation performed by the following piece of code ?

```
MVI A, 07h
RLC
MOV B, A
RLC
RLC
ADD B
```

12. (*Gaonkar*, Chapter 9)

```
2000 LXI SP, 2100h  delay:  2064 PUSH H
2003 LXI B, 0000h           2065 PUSH B
2006 PUSH B                 2066 LXI B, 80FFh
2007 POP PSW        loop:   2069 DCX B
2008 LXI H, 200Bh           206A MOV A, B
200B CALL 2064h             206B ORA C
200E OUT 01h                206C JNZ LOOP
2010 HLT                    206F POP B
                            2070 RET
```

(a) What is the status of the flags and the contents of the accumulator after the execution of the POP instruction located at 2007*h* ?

(b) Specify the stack locations and their contents after the execution of the `CALL` instruction (not the CALL subroutine).

(c) What are the contents of the Stack Pointer register and the Program Counter after the execution of the `CALL` instruction ?

(d) Specify the memory location where the Program Counter returns after the subroutine.

(e) What is the final fate of the program ?

13. **The `CALL` of the Wild** Show the timing diagram with T-states corresponding to the following instruction at memory location 2040*h*: `CALL` 2070*h*. The opcode for `CALL` is 'CD'. Assume that SP initially contains 2400*h*. At each stage, show the following signals: ALE, A_{15} - A_8, AD_7 - AD_0, {IO/$\overline{M}$, S_1, S_0 }, $\overline{RD}$, $\overline{WR}$, and the contents of the following registers: PC, SP and the WZ (internal) register pair.

14. **The `RET`urn of the Dragon** Repeat the above excercise for the corresponding unconditional return statement `RET` at the end of the subroutine.

15. **'Sort of Difficult' ...** Implement the BUBBLESORT sorting algorithm for a list of unsigned characters (bytes) in memory, along with the total number of memory bytes to be sorted, in ascending order.

16. **IP Addresses, Ethernet** Network communication is in the form of data *packets* of bytes. Each machine on the network receives all packets, irrespective of whether the packet is meant for the machine, or not. Write an 8085 program segment with an infinite loop, to check is a packet is meant for that machine. The IP address of a machine is a 4-byte number, such as `10.107.1.2`. Suppose that the 8085-based machine stores it's IP address in (fixed) memory locations 2040h - 2043h. The program has an infinite loop: it calls subroutine `get_packet` (this places the packet in consecutive memory locations starting from 3000h, with the first 4 bytes containing the IP address of the machine the packet is meant for). If the IP address of the packet matches the IP address of the machine, the system calls subroutine `process_packet`. Assume the availability of subroutines `get_packet` and `process_packet`. Write code to perform the above operation.

17. **Recursive Directory Listing** Write a program to recursively list the files in all subdirectories, starting from a given directory. Subroutine `get_info` (assume this to be given to you) returns in the HL pair, a pointer (memory address) to a 16-byte memory block, which contains information about a directory entry. The first byte is 77h if it is a file, and FFh if it is a subdirectory. The first call to `get_info` returns information about the first entry in the current directory, and so on. If `get_info` is called after the last entry in the current directory has been accessed, the subroutine returns a pointer to a block of 16 zero bytes (the pointer may be non-zero).

18. **Sort of Minimally Difficult ...** First, write an $\mathcal{O}(n)$ subroutine `min` with the following specifications: This finds out the minimum number in a set of n numbers stored consecutively in memory (the numbers as well as n are each 1 byte-long) - the resulting array has the same numbers, except that the minimum is at the first location. The parameters to the routine should be the address of the location of the first number in the array in the DE pair, and n in the register C. This routine should not modify the DE pair, but may modify any other register. Now, write a subroutine `selection_sort`, which will take parameters: the address of the first number in the array in the HL pair, and the number of elements (n) in register C. This should sort the n numbers, using routine `min` in the process. You can assume that the stack has been appropriately initialized.