New Commands to Improve the Monitor for the ELWE Microcomputer

Dr. Afaf Badie Jamil
Department of Computer Science- College of Education for Women
Baghdad University

Abstract
The base of the ELWE MPF-1 microcomputer is the Z80 microprocessor. The Z80 programs are written in assembly language. The main advantage of assembly language is that: it is much faster to code and the mnemonics makes it easier for the user to remember the instruction.
The purpose of this paper is to improve the monitor process for didactic microcomputer ELWE MPF-1 by appending new powerful commands (MOVE, FILL, SEARCH and COMPARE) to the existing monitor to make it more useful and flexible. 8085 assembly language is used to execute this program. The letters used for abbreviation: M for MOVE, F for FILL, S for SEARCH and C for COMPARE.

ايعازات جديدة لتحسين المراقب ل ELWE مايكروكومبيوتر

د. عفاف بديع القدو
جامعة بغداد/كلية التربية للبنات/قسم الحاسبات

الخلاصة
إن الأساس للمايكروكومبيوتر ELWE MPF-1 هو المعالج Z80. إن برامج Z80 مكتوبة بلغة التجميع. الفائدة الأساسية للغة التجميع هي سرعة كتابة الشفرة والمختصرات جعلتها أسهل للمستخدم لتذكر العبارات.
الهدف من هذا البحث هو تحسين المراقب الخاص بالمايكروكومبيوتر ELWE وذلك بإضافة ايعازات قوية جديدة هي (MOVE, FILL, SEARCH, COMPARE) للمراقب الأصلي الموجود وذلك لجعله أكثر قوة ومرونة. استخدمت لغة التجميع 8085 لتنفيذ هذا البرنامج. الأحرف المستخدمة هي: M للإيعاز MOVE والحرف F للإيعاز FILL والحرف S للإيعاز SEARCH والحرف C للإيعاز COMPARE.

1. Introduction

The Z80 is a CPU - central processing unit - having the ability to fetch and execute machine language instructions. These instructions, in turn, can specify simple operations such as transferring an item of data between the outside world and one of the Z80's internal registers, or performing simple computational operations (e.g.addition). The ELWE MPF-1 is a COMPUTER - it includes a Z80 CPU, plus a memory system, plus an input-output system [6].

1.1 Z80 CPU Architecture

A. The Z80 is a pure 8-bit microprocessor, which means that its internal registers and data paths (as well as its external data bus) are 8 bits wide [5][8][10].

1. This means that basic arithmetic operations are performed on 8-bit operands, which can represent values in the range of 0..255 (unsigned) or -128 .. 127 (signed).
2. When it is desired to work on numbers having a larger range of values, it is possible to combine two 8-bit operands to yield a 16-bit value (range 0..65535 (unsigned) or -32768..32767 (signed) or even to combine four to yield a 32-bit value. However, the internal arithmetic is done 8-bits at a time - thus adding two 16-bit numbers requires two steps, one for each half of the operands.
3. Many modern CPU's are 32-bit processors, which means their internal data paths and registers are 32 bits wide (though some high-end processors are 64-bit). These machines typically allow the user the option of working with 8, 16, or 32 bit operands, so less storage can be used when a smaller range of values is all that is needed. (e.g. character strings are typically represented by using 8 bits per character).
4. However, 8-bit CPU's are still manufactured and used extensively in embedded systems - e.g. home appliances - where the power of a 16-bit or 32-bit CPU is not needed and does not warrant the extra cost and wiring complexity. (The latter is a key issue - a 32-bit system needs four times as many data lines between the CPU and memory as 8-bit system needs, which add to manufacturing complexity and cost. It is probable that a typical home contains more 8-bit CPU's than anything else!)

B. The Z80 CPU connects to the outside world through two BUSES plus a set of control lines Figure (1),[9].

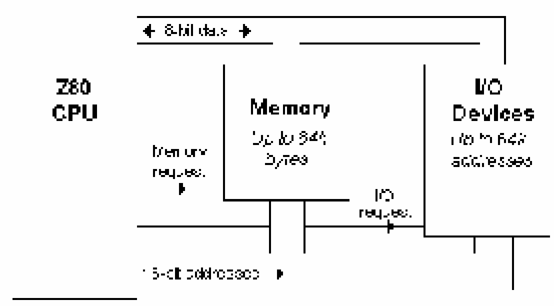

Figure (1): Z80 microprocessor 8-bit Data and 16-bit Addresses

1. The 8-bit DATA BUS can be used to transfer a byte of information between the CPU and the outside world.
2. The 16-bit ADDRESS BUS is used to specify exactly where the data is to be transferred from or to. With 16-bits, it is possible to specify 65536 (64K) unique addresses - thus the Z80 can connect to up to 64K of memory.
3. Control lines are used to specify what type of operation is to be performed - e.g. read (transfer of data to the Z80 from some external source) or write (transfer of data from the Z80 to some external destination).
4. On the MFP-1, these buses are connected to on-board memory chips that provide 8K bytes of read only memory (ROM) and 4K bytes of read- write memory (RAM), plus various IO devices (keyboard, display, etc.).

C. The Z80 is basically a one-accumulator machine, which means that for most arithmetic and logical operations the A register (the accumulator) contains one of the source operands and receives the result of the operation. This contrasts with many newer computers, which have a number of general registers which are equally capable of participation in arithmetic and logical operations.

D. The Z80 has a total of 16 8-bit registers and 4 16-bit registers [5][7][9] Figure (2).

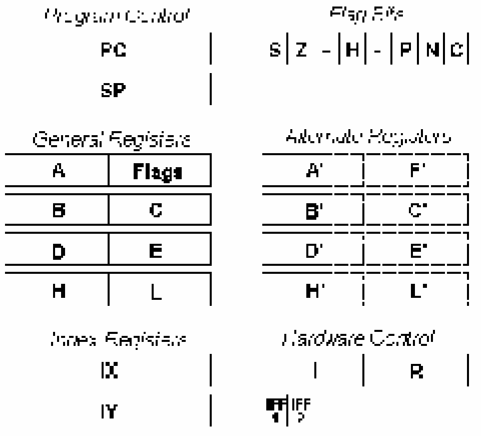

Figure (2): Z80 Processor Registers

In Figure (2), the following are found:-

1. The 16-bit PC contains the address of the next instruction to be executed.
2. The 16-bit SP is a stack pointer used to maintain a hardware stack in memory - it contains the address of the memory cell holding the top item on the stack.
3. The 8-bit A register is the accumulator, and participates in most arithmetic and logical operations.
4. The 8-bit registers B, C, D and E provide temporary storage for intermediate results of operations, and can also be paired up (BC, DE) to form 16-bit registers that can be used as pointers to memory cells.
5. The 8-bit registers H and L are seldom used as 8-bit registers. More often they are paired to form a 16-bit register that is used to point to a memory location. Many memory-reference instructions require that the address of the item to be fetched or stored be in HL.

6. The processor also contains two index registers (IX and IY) that can be used to address data in memory.

2. New Commands

a- MOVE Command: is for transfer a block of data from a specific area addressed by addr1 and addr2 of memory to another area addressed by addr3. The syntax of it:

<M>= addr1 addr2 addr3

The MOVE command can transfer data if source and destination areas of memory are overlapping as shown in Figure 3:

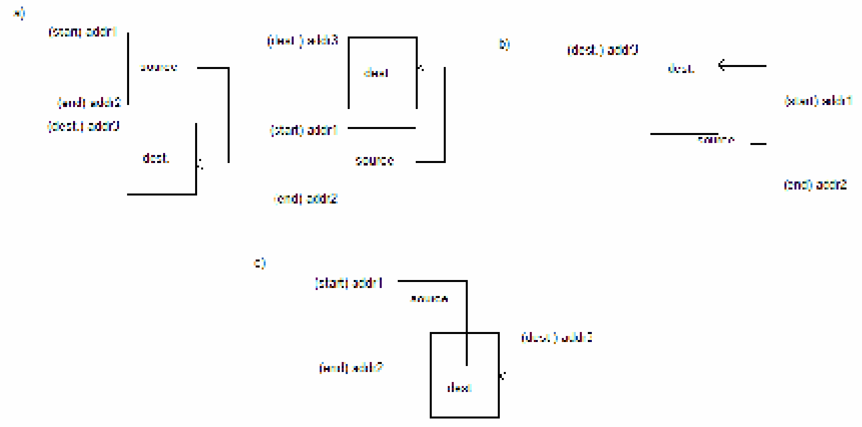

Figure (3): Cases of Overlapping

a- Ordinary transfer.

b- Overlap (move up).

c- Overlap (move down).

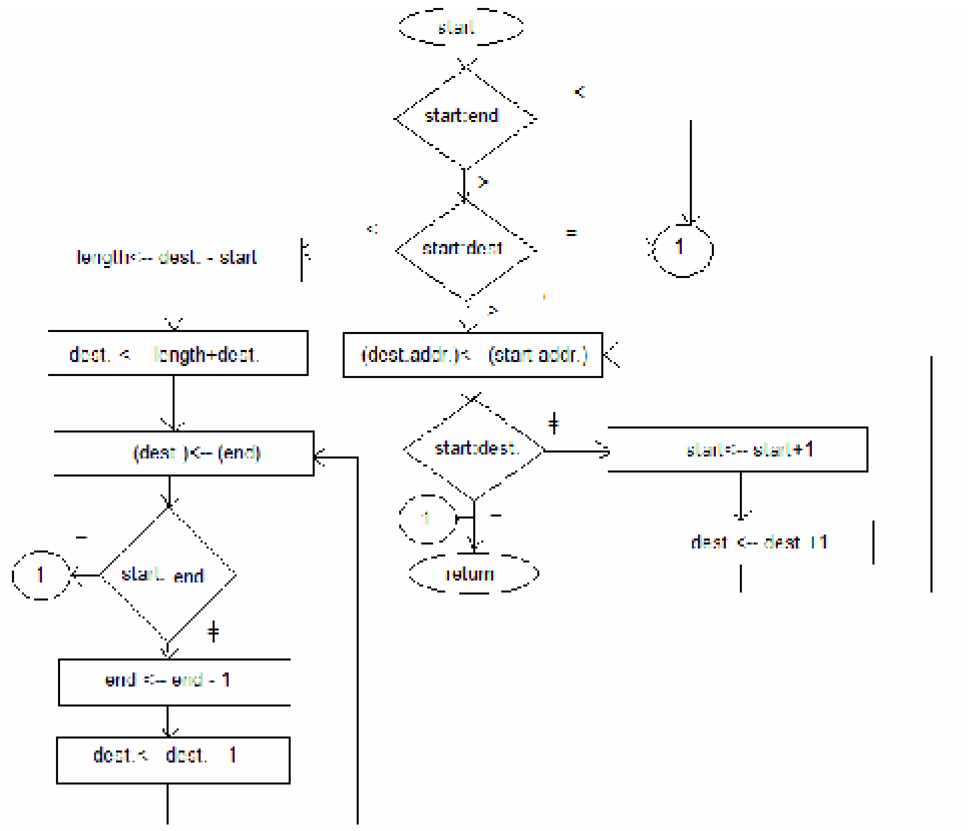

Figure (4): MOVE Command Subprogram Flowchart

<u>Example1:</u>
The following state of memory is given:

F800	7D	F900	00
F801	91	F901	00
F802	6F	F902	00
F803	7C	F903	00
F804	98	F904	00

After execution of the MOVE command:
<M>=F800 F804 F900 the content of memory will be changed:

F800	7D	F900	7D
F801	91	F901	91
F802	6F	F902	6F
F803	7C	F903	7C
F804	98	F904	98

You cannot execute this program if it contains JUMP instruction because all addresses will be changed, except if it contains JUMP RELATIVE instruction instead of JUMP. All the time end address must be greater than start address otherwise the program is terminated.

b- FILL Command: is for storing a word (two bytes) of data in a specified area of memory, addressed by addr1 and addr2. its syntax of is:

<F>=addr1 addr2 word

Example2:
The following state of memory is given:

```
F800  00
F801  00
F802  00
F803  00
F804  00
```

After execution of the FILL command:
<F>=F800 F804 FF00   the content of memory will be changed:

```
F800  FF
F801  00
F802  FF
F803  00
F804  FF
```

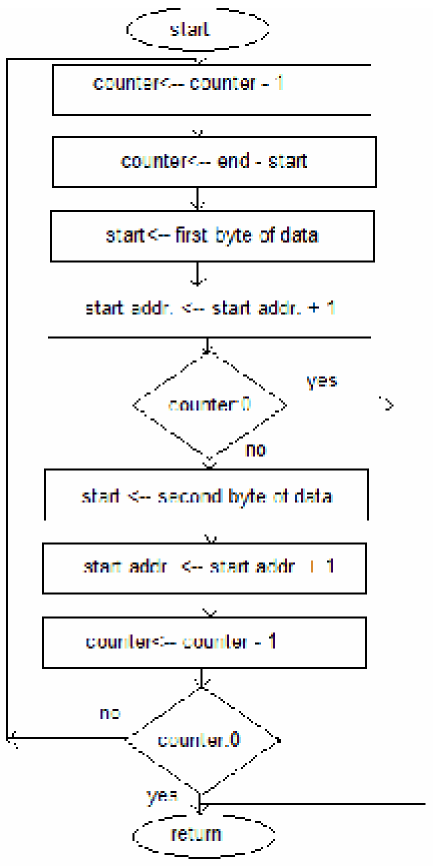
start
counter<-- counter - 1
counter<-- end - start
start<-- first byte of data
start addr. <-- start addr. + 1
counter 0
yes
no
start <-- second byte of data
start addr <-- start addr + 1
counter<-- counter - 1
no
counter 0
yes
return

Figure (5): FILL Command Subprogram Flowchart

c- SEARCH Command: is used to find all contents of specified area of memory addressed by addr1 and addr2, contents of these area must be equal to specified byte1 of word masked by byte2. its syntax of is:

<S>=addr1 addr2 word

Example3:

The following state of memory is given:

FC00	91
FC01	58
FC02	91
FC03	58
FC04	FB

After execution of the SEARCH command:

<S>=F800 F804 FF04

```
FC00  91
FC01  58
FC02  91
FC03  58
FC04  FB*
```

The operation of searching is done like this:

```
        1111 1111            FF
XOR     0000 0100       XOR  04
------------------------   -------------
        1111 1011            FB


       1001 0001                 91
AND  1111 1011            AND  FB
-----------------------     ---------------
        1001 0001                91
```

At the beginning XOR is made between FF masking byte and 04 searching byte the result FB is using for comparing with bytes from the specified area of memory by making AND operation. If the result is different from zero such a location of memory will not be displayed.

The result 91 is different than zero the address FC00 and contents of this location will not be displayed. The rest is done in similar way. The address FC04 and contents of this location will be displayed as shown below:

```
        1111 1111            FF
XOR     0000 0100       XOR  04
------------------------   -------------
        1111 1011            FB


       1111 1011                 FB
AND  1111 1011            AND  FB
-----------------------     ---------------
        0000 0000                00
```

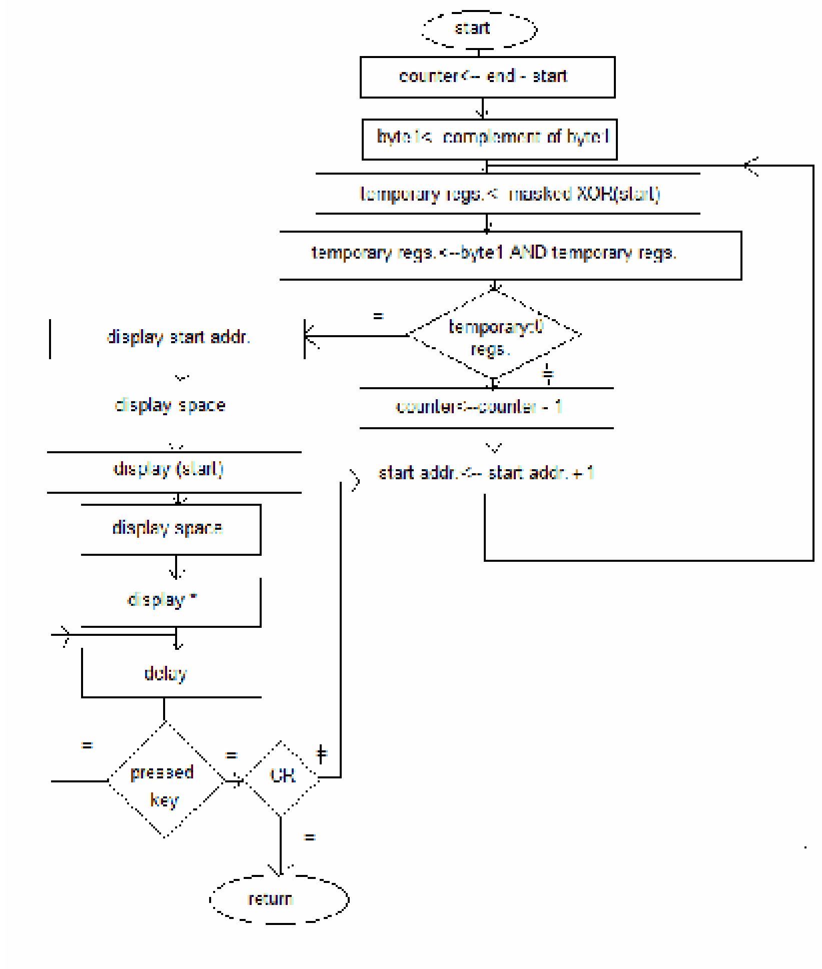

Figure (6): SEARCH Command Subprogram Flowchart

d- COMPARE Command: is for comparing two areas of memory, specified by addr1 and addr2 to the addr3, and finding if the contents of these areas are equal or not. its syntax is:

<C>=addr1 addr2 addr3

Example4:
The following state of memory is given:

F800	FF	F900	00
F801	C9	F901	00
F802	CF	F902	00
F803	00	F903	00

```
F804   F9              F904   F9
```

After execution of the COMPARE command:

```
<C>=F800 F804 F900
```

The result is:

```
F800   FF#F900   00   %      press any key to continue
F801   C9#F901   00   %      press any key to continue
F802   CF#F902   00   %      press any key to continue
F803   00=F903   00   %      press any key to continue
F804   F9=F904   F9   %
```

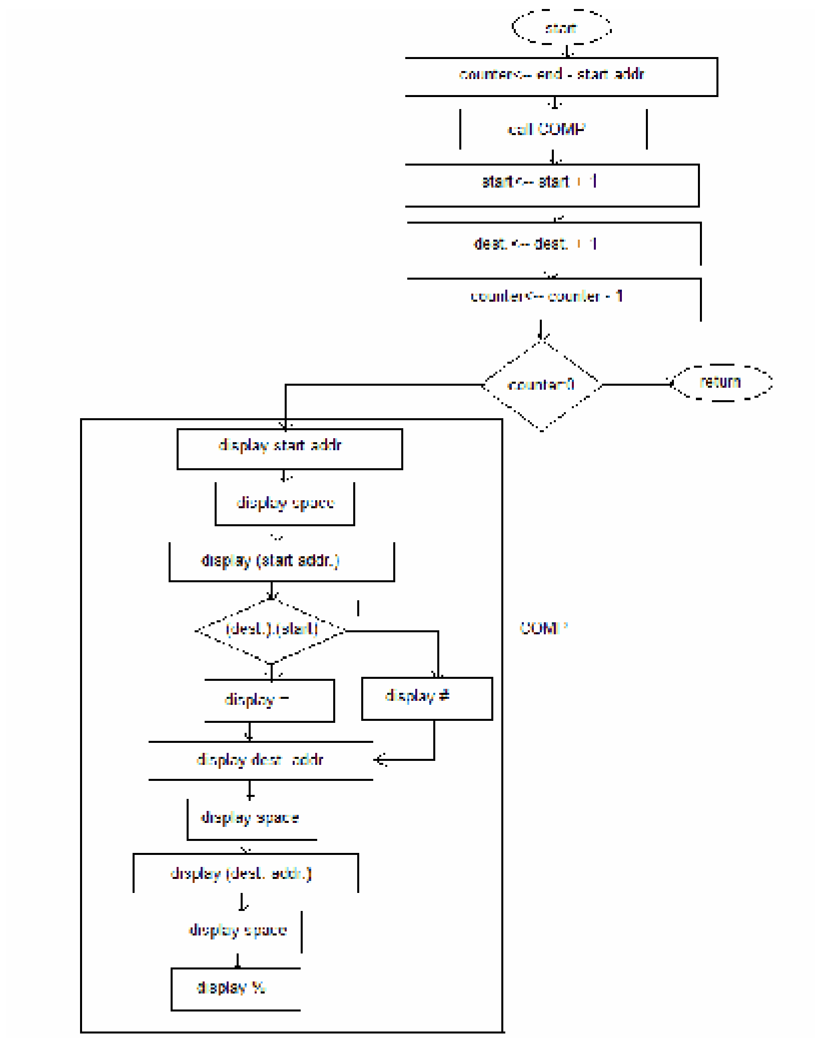

Figure (7): Flowchart of COMPARE Command

3. Main loop of monitor program [1][2][3][4]

These two flowcharts below explain the Z80 main loop of ELWE MONITOR program before and after modification.

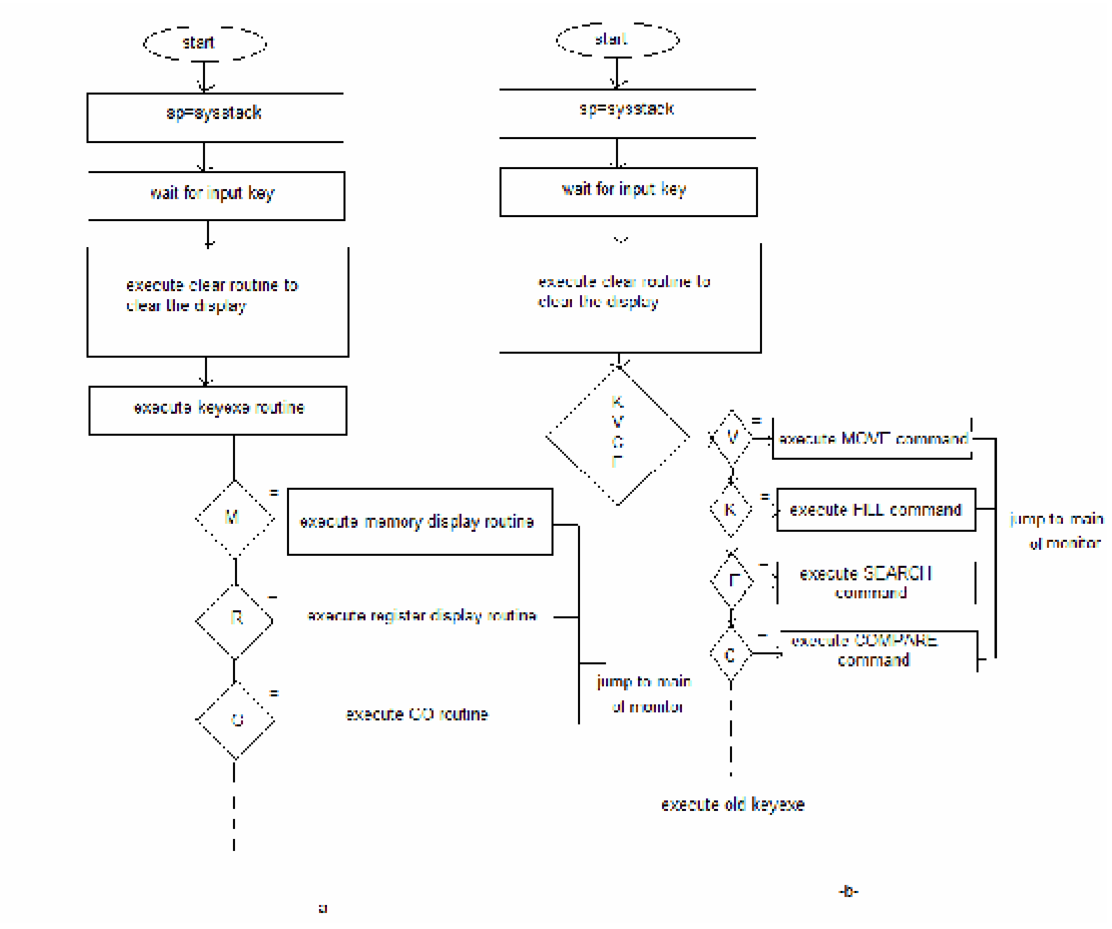

Figure (8): a- Old Keyexe Subroutine Flowchart.
b- New Keyexe Subroutine Flowchart.

If we compare the old and the new ELWE monitor subroutine called KEYEXE it is easy to find the difference: the new letters C,K,E, and V are connected with the new commands for the ELWE microcomputer. If processor has recognized one of these letters a JUMP to the proper subprogram is made otherwise the execution of one of old subprogram started.

The function of the KEYEXE subroutine is to check the pressed key and then execute the proper subprogram according to the letter and finally return to main loop of ELWE program.

4. Connection of the new KEYEXE subroutine

The connection to the monitor is done by changing only two bytes of the main loop of ELWE monitor program. The below flowchart explains this connection:

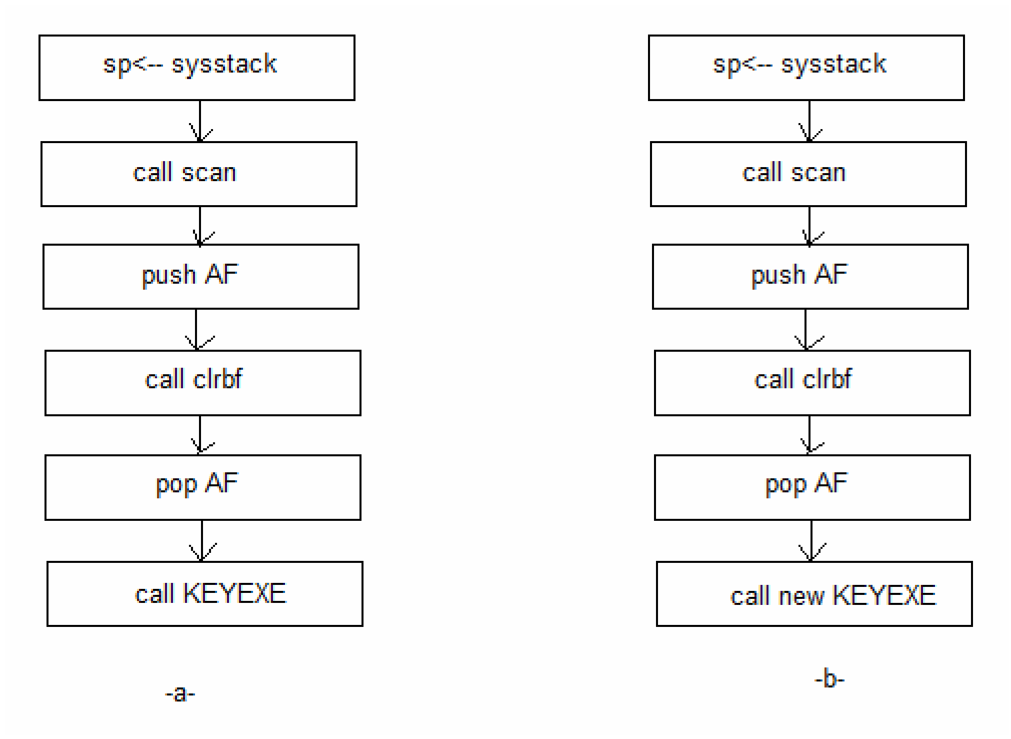

Figure (9): Main Loop of ELWE Monitor Program
a-Before Modification
b-After Modification

The new KEYEXE contains both letters of new and old KEYEXE subroutine.

5. Conclusions

These commands were selected for the implementation since these are useful to use the notion By adding these new commands the monitoring problem will solved.

References

[1] Crane J., "Laboratory Experiments for Microprocessor System", Prentice Hall, Englewood, 1980.
[2] MPF-IP Monitor program source listing, Multitech Industrial Corp., 1983.
[3] MPF-IP User's Manual, Multitech Industrial Corp., 1983.
[4] Philips I&E, Ei idhoven, "PMDS-II Debugger Manual, PMDA System Reference Documentation", the Netherlands, 1985.
[5] Russel C. Bjork, "An Example of the Z80", 1998. www.galia.fc.uaslp.mx
[6] Steve C., "Build Your Own Z80 Computer", Byte Books, 1981.
[7] Steve Ciareia, "Design Guide Lines and Application Notes", 1981.
[8] Thomas Scherrer, "Home of the Z80 CPU Family", 2011. www.Z80.info
[9] The Z80 Microprocessor.mht, 1998. www.penguicon.sourceforge.net
[10] Z80 Microprocessor Specifications eHow_com.mht, 2011. www.eHow.com