1. FETCH - Retrieving the relevant instruction from a specific memory location.

2. DECODE - Understanding the mechanism the fetched instruction invokes.

3. EXECUTE - Carrying out the required steps

This cycle continues repeatedly inside the Central Processing Unit (CPU) until all the instructions are successfully executed. Usually in a simple processor these steps occur sequentially, finishing one Execution before the next Fetch occurs. The breakdown of each of the instructions are expressed below.

• **FETCH**

The current address in the program counter (PC) points to address of the next instruction to be fetched. This instruction is fetched from Instruction ROM and stored in the Memory Buffer Register Unit (MBRU).At the end of each FETCH cycle, the PC is incremented by one representing the next instruction to be fetched. FETCH is a two-step state machine.

FETCH1 MBRU ← IRAM[PC]; FETCH

FETCH2 PC ← PC + 1

• **DECODE**

To invoke the correct execution cycle, the processor has to differentiate between the instructions that are being fetched. The MBRU gives the fetched instruction to the Control Store and the output of the control store is the corresponding control signal. If the instruction is of one state, the next FETCH cycle is commenced.

**• EXECUTE**

1. **NOP**

No operation performed. This operation is utilized when a waiting cycle is required in order to have some processed data available at an end point.

1. **LDAC**

The data in the DRAM with the location pointed by the current address in the MAR is read and loaded to MDR. Then that data is loaded to AC. The next FETCH cycle will commence afterwards.

LDAC1 MDR ← DRAM[MAR]; READ

LDAC2 AC ← MDR

1. **STAC**

The data in AC is loaded to MDR. Next this data is written to the location in DRAM provided by the address in MAR. Then the next FETCH cycle begins.

STAC1 MDR ← AC

STAC2 DRAM[MAR] ← MDR;WRITE

1. **CLAC**

Zero is assigned to the AC. (AC is cleared.) The zero flag is indicated. The next FETCH cycle begins afterwards.

CLAC AC ← 0;

1. **MVACR1, MVACR2, MVACR3, MVACL, MVACE**

Moving the value in the AC to the specified register is the basic operation of all of the above Instructions. The next FETCH cycle begins after each of these single state instruction.

MVACMAR MAR ← AC

MVACR1 R1 ← AC

MVACR2 R2 ← AC

MVACR3 R3 ← AC

MVACL L ← AC

MVACE E ← AC

1. **MVR1AC, MVR2AC, MVR3AC, MVEAC, MVLAC**

Moving the value stored in the specified register to AC is the basic function in these instructions. The next FETCH cycle begins after each of these single state instruction.

MVC1 AC ← C1

MVC2 AC ← C2

MVC3 AC ← C3

MVL AC ← L

1. **INAC**

Incrementing the value in AC by one occurs in this instruction. The next FETCH cycle begins after this single state instruction.

INAC AC ← AC + 1

1. **DCAC**

AC is decremented by one in this instruction. next FETCH cycle commences after this step.

DCAC AC ← AC - 1 ;

1. **ADD256**

In both of these instructions, the value in the specified register is added to the value in the AC currently and loaded to AC itself. The next FETCH cycle begins after this single state operation.

ADD256 AC ← AC + L

1. **ADDR3**

ADDR3 AC← AC + R3

1. **SUB256**

SUB256 AC ← AC - L

1. **SUBL, SUBE**

In both of these instructions, the value in the specified register is subtracted from the value in the AC currently and loaded to AC itself. The next FETCH cycle begins after this single state operation.

SUBL AC ← AC - L ; IF (AC-L) == 0 THEN Z = 1 ELSE Z = 0

SUBE AC ← AC - E ; IF (AC-E) == 0 THEN Z = 1 ELSE Z = 0

1. **DIV2, DIV16**

The binary equivalent of dividing by 16 is shifting the value to the right by 4 digits. In this case the value in the AC is divided by 16 and then the next FETCH cycle begins.

DIV2 AC ← AC>> 1

DIV16 AC ← AC>> 4

1. **MUL2, MUL4, MULL**

The binary equivalent of multiplying by 2 is shifting the value to the right by 1 digit. In these cases the value in the AC is multiplied by 2, 4 and 256 by relevant shifts and then the next FETCH cycle begins.

MUL2 AC ← AC<< 1

MUL4 AC ← AC << 2

MULL AC ← AC <<8 (256)

1. **JMPZ**

In this case the zero flag is checked and if Z=0 (i.e. AC is not zero), then no jumping occurs so the PC is incremented by one. In the case of Z being 1, the three states similar to that of JUMP instruction proceeds. At the end the next

FETCH cycle will occur at the branched location.

1. **JMPZY**

JMPZY1 (Z=0) PC ← C

JMPZY2 (Z=0)MBRU ← IRAM[PC] ; FETCH

JMPZY3 (Z=0) C ← MBRU

1. **JMPZ**

JMPZX1 (Z=1) PC ← PC + 1

1. **JMPN**

In this case the zero flag is checked and if Z=0 (i.e. AC is not zero), then no jumping occurs so the PC is incremented by one. In the case of Z being 1, the three states similar to that of JUMP instruction proceeds. At the end the next

FETCH cycle will occur at the branched location.

1. **JMPNY**

JMPNY1 (Z=0) PC ← C

JMPNY2 (Z=0) MBRU ← IRAM[PC] ; FETCH

JMPNY3 (Z=0) C ← MBRU

1. **JMPNX**

JMPNX1 (Z=1) PC ← PC + 1