**FETCH, DECODE, EXECUTION CYCLE**

**Fetching instructions**

Fetch cycle consists of 4 states. Fetch cycle is run by the state machine with FETCH1 being set as the next state at the beginning.

FETCH1: Fetch

FETCH2: MBRU  IRAM

FETCH3: PC  PC+1

// FETCH4: Next State  MBRU

**Decoding**

Decode of instructions is the next task of the CPU after fetching instructions from the instruction memory. The CPU has to differentiate between the instructions fetched from the instruction memory in order to invoke the correct execution cycle. This task is done by the state machine or the control unit of the processor. Memory Bus Register Unsigned (MBRU) inputs the fetched instruction to the control unit and the control unit decodes the instruction to output the control signals of the relevant state followed by the next states of the instruction or returns to fetch cycle if the instruction has only one state.

**Executing**

**NOP**

NOP instruction is used to do nothing. This instruction is useful to skip a clock cycle or two in order to wait until the data is ready at the end point.

**LDAC Instruction**

This instruction consists of two states.

LDAC1: read

LDAC2: MDR  DRAM [MAR]

1st state sends the read signal to the data memory and the MDR. This state involves loading the data from the address given in the previous state (AR) to AC. Then CPU moves to the fetch routine.

**LDACV Instruction**

This instruction consists of two states.

LDACV1: fetch

LDACV2: MBRU  IRAM [PC]

LDACV3: PC  PC+1

LDACV4: AC  MBRU

LDACV5: AC  AC<<8

LDACV6: fetch

LDACV7: MBRU  IRAM [PC]

LDACV8: PC  PC+1

LDACV9: AC  AC+MBRU

1st state sends the read signal to the data memory and the MDR. This state involves loading the data from the address given in the previous state (AR) to AC. Then CPU moves to the fetch routine.

**STAC Instruction**

This instruction consists of 2 states.

STAC1: MDR  AC

STAC2: write

This involves copying the contents of the AC to the memory address in the data memory pointed by the MAR. Write data memory address should be available in the MAR. Move the data from the address stored register to the MAR register.

**MVACR1, MVACR2, MVACR3, MVACR4, MVACR and MVACMAR Instructions**

These instructions consist of only one state. The CPU copies the contents of the AC to R1, R2, R3, R or MAR and moves to fetch routine.

MVACR11: R1AC

MVACR21: R2AC

MVACR31: R3AC

MVACR41: R4AC

MVACR1: RAC

MVACMAR: MARAC

**MOVR, MOVR1, MOVR2, MOVR3 and MOVR4 Instructions**

These instructions involve only one state. The CPU copies the contents of the R1, R2, R3 or R to AC and moves to fetch routine.

MOVR1: ACR1

MOVR2: ACR2

MOVR3: ACR3

MOVR4: ACR4

MOVR: ACR

**JUMP Instruction**

If z flag equals to zero three states are involved. In the 2nd state instruction in the memory address is loaded into AC. Then value of AC is copied to PC. Then the CPU moves back to fetch routine.

JUMP1: fetch

JUMP2: MBRU  IRAM [PC]

JUMP3: PC  PC+1

JUMP4: AC  MBRU

JUMP5: AC  AC<<8

JUMP6: fetch

JUMP7: MBRU  IRAM [PC]

JUMP8: PC  PC+1

JUMP9: AC  AC+MBRU

JUMP10: PCAC

**JMPZ Instruction**

If z flag equals to 1, PC is incremented by 1 and the CPU move to fetch routine and start to fetch the next instruction to be executed.

If Z=0;

JMPZN1: PCPC+1

JMPZN2: PCPC+1

If z flag equals to zero three states are involved. In the 2nd state instruction in the memory address is loaded into AC. Then value of AC is copied to PC. Then the CPU moves back to fetch routine.

If Z=1;

JMPZY1: fetch

JMPZY2: MBRU  IRAM [PC]

JMPZY3: PC  PC+1

JMPZY4: AC  MBRU

JMPZY5: AC  AC<<8

JMPZY6: fetch

JMPZY7: MBRU  IRAM [PC]

JMPZY8: PC  PC+1

JMPZY9: AC  AC+MBRU

JMPZY10: PCAC

**JMNZ Instruction**

If z flag equals to 1, PC is incremented by 1 and the CPU move to fetch routine and start to fetch the next instruction to be executed.

If Z=1;

JMNZY1: PCPC+1

JMNZY2: PCPC+1

If z flag equals to zero three states are involved. In the 2nd state instruction in the memory address is loaded into AC. Then value of AC is copied to PC. Then the CPU moves back to fetch routine.

If Z=0;

JMNZN1: fetch

JMNZN2: MBRU  IRAM [PC]

JMNZN3: PC  PC+1

JMNZN4: AC  MBRU

JMNZN5: AC  AC<<8

JMNZN6: fetch

JMNZN7: MBRU  IRAM [PC]

JMNZN8: PC  PC+1

JMNZN9: AC  AC+MBRU

JMNZN10: PCAC

**INAC and DEAC Instructions**

INAC1: ACAC+1

INAC Instruction involves the CPU to add 1 to the contents of AC and write back to AC.

DEAC1: ACAC-1

This instruction is straightforward which involves subtracting 1 from the contents of AC and writing back to AC.

After the states described above, the CPU moves to the fetch routine.

**SUB and ADD Instructions**

SUB1: AC AC-R

SUB instruction relates to subtracting the contents of R from AC and writing back to AC.

ADD1: ACAC+R

ADD instruction adds the contents of R to AC and writes back to AC.

Then the CPU moves to fetch routine and starts fetching the next instruction from the instruction memory.

**ADDV Instruction**

This instruction consists of 2 states which add a value resides in the immediate memory location in IRAM to AC and writes back to AC.

ADDV1: R  AC

ADDV2: fetch

ADDV3: MBRU  IRAM [PC]

ADDV4: PC  PC+1

ADDV5: AC  MBRU

ADDV6: AC  AC<<8

ADDV7: fetch

ADDV8: MBRU  IRAM [PC]

ADDV9: PC  PC+1

ADDV10: AC  AC+MBRU

ADDV11: ACAC+R

In the next state the value in the memory location which is next to “ADDV” is added to AC and written back to AC.

**SUBV Instruction**

This instruction consists of 2 states which add a value resides in the immediate memory location in IRAM to AC and writes back to AC.

SUBV1: R  AC

SUBV2: fetch

SUBV3: MBRU  IRAM [PC]

SUBV4: PC  PC+1

SUBV5: AC  MBRU

SUBV6: AC  AC<<8

SUBV7: fetch

SUBV8: MBRU  IRAM [PC]

SUBV9: PC  PC+1

SUBV10: AC  AC+MBRU

SUBV11: ACAC-R

The value in the AC register is subtracted from the value in the memory location which is next to “SUBV” and written back to AC.

**DIV and MUL2, MUL4, MUL512 Instructions**

These 2 instructions consist of one state which divides the contents of AC by 2 and multiplies by 4 then writes back to AC.

DIV1: AC  AC>>1

MUL21: AC  AC<<1

MUL41: AC  AC<<2

MUL5121: AC  AC<<8

MUL5122: AC  AC<<1

**INR1 and INR2 Instructions**

These 2 instructions consist of one state which divides the contents of AC by 2 and multiplies by 4 then writes back to AC.

INR11: AC  R1

INR12: AC  AC+1

INR13: R1  AC

INR21: AC  R2

INR22: AC  AC+1

INR23: R2  AC

**CLAC Instruction**

CLAC1: AC0

The CLAC instruction can be executed by one state. It involves the CPU to clear the contents of the accumulator and start fetching the next instruction from the instruction memory.