

# A-MISC: The Arabic Medium Instruction Set Computer Architecture Design

Asbahiya A. Abu-Samra

**Abstract**—The design and development of 16-bit Medium Instruction Set Computer (A-MISC) covering the performance of Complex Instruction Set Computers (CISC) and Reduced Instruction Set Computers (RISC) is proposed here. The instruction set of the proposed A-MISC computer includes Memory Reference Instructions with direct, indirect and immediate addressing features, Register Reference instructions dealing operations within the registers of the CPU, IO reference instructions concerning the peripherals, Stack instructions controlling the stack and some other miscellaneous instructions. To have a reduced complexity for the hardware the data word size is set as a standard of 16 bits. The lengths of instructions are of either one word or 2 words depending upon the category of instructions. The address of the memory is denoted by 24 bits resulting in physical memory capacity of 16M words for the computer. Most essential flags available in the CISC computers are included in the A-MISC. The control unit is designed on hardwired concept. For the time being the programs for A-MISC are written in symbolic assembly program. The Assembler package for this computer is developed with Borland C and it would run two passes. Sample programs are run on the computer and the performance of the computer is estimated.

**Index Terms**—A-MISC, Computer Architecture, Control unit, Assembler .

## 1. INTRODUCTION

Complex Instruction Set Computers (CISC) are being widely used in practice and in several instances Reduced Instruction Set Computers (RISC) are also employed. Although the objectives are nearly same the application ranges and realization aspects are bit different amongst these computers. Some applications in modeling and simulation areas could be handled effectively in computers which have drawn the advantages of CISC and RISC computers. In designing such computers several architectures were adapted in the past and varieties of computers were developed and used in practice. CISC computers such as PCs and work stations developed using processors like Pentium are the most extensively used computers in practice. Owing to its high speed of operation Reduced Instruction Set Computers are being used for some specific applications. Combining the features of RISC and CISC a special architecture is being proposed here. As the size of the instruction set is in between the CISC and the RISC the proposed architecture is named as A-MISC.

Asbahiya A. Abu-Samra is a lecturer at College of computer and information sciences, King saud university, Saudi Arabia ( phone: + 966-502-444360 ; e-mail: [asbahiya@hotmail.com](mailto:asbahiya@hotmail.com); [aabusamra@ksu.edu.sa](mailto:aabusamra@ksu.edu.sa)).

## 2. REGISTER STRUCTURE

In the architecture of the A-MISC, in addition to the conventional registers such as PC(Program Counter), AR(Address Register) and SP(Stack Pointer) of 24-bits size, the register structure includes user accessible registers of 16-bits word-size named as A, B, C and D. The data word-size for processing is uniformly kept as 16-bits and therefore the registers holding the data are of 16-bits size. The address holding registers such as PC,SP and AR hold the full address of the memory and registers like instruction pointers and index registers are not included here. Therefore the register structure is simplified in A-MISC as compared to RISC computers and even the CISCs.

## 3. INSTRUCTION SET

The instruction set of the proposed A-MISC computer includes Memory Reference Instructions with direct, indirect and immediate addressing features, Register Reference instructions dealing operations within the registers of the CPU, IO reference instructions dealing the peripherals, Stack instructions controlling the stack and some other miscellaneous instructions.

### Instruction Format

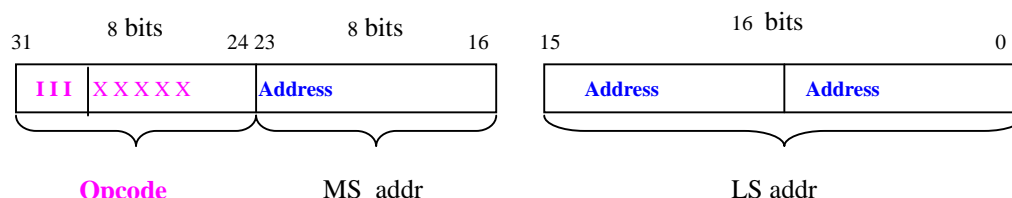
The instruction word is of one word of 16 bits or two 16-bits words and the data word size is always 16 bits. Memory reference and immediate addressing instructions are 2 words of length. Table 1 gives the types of instructions used in A-MISC. The type of the instructions is represented by the most significant three bits denoted as III. The instruction format for some type of instructions are shown in Fig.1.

### Memory Reference Instructions

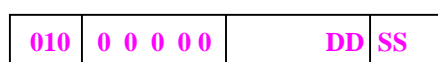
Table 2. gives different memory reference instructions used in A-MISC. Incidentally, the assembly instruction for adding the contents of memory to A register is denoted by ADD addr where it occupies least significant 24 bits in the instruction code. The most significant three bits (III) of the instruction code of the first word denote the type of instruction and the first code 000 specifies the memory reference instruction with direct addressing. The next 5 most significant bits specify the opcode doing the type of arithmetic or logic operation concerning the memory reference instruction.

Table 1. Instruction Types and Codes

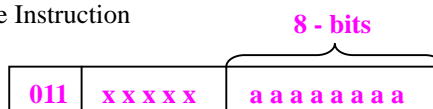
Code Bits I I I	Instruction selected	No. of Words
0 0 0	Mem Ref- direct addressing	2
0 0 1	Mem Ref-Indirect addr (Ms 8-bits of addr are same)	2
0 1 0	Register Ref-Instruction.	1
0 1 1	I/O Ref-Instruction.	1
1 0 0	Stack Instruction.	1
1 0 1	Immediate Addressing	2
1 1 0	Vector Processing (Future)	2
1 1 1	Future	



a. Memory Reference Instruction



b. Register reference instruction



c. I/O Reference Instruction

Fig.1. Instruction Word Format

Table 2. Memory Reference Instructions

Symbol	Direct	Indirect	Description
	Hex code	Hex code	
ADD addr	0 0 x x x x x x	2 0 x x x x x x	$A \leftarrow A + \text{mem}$
ADC addr	0 1 x x x x x x	2 1 x x x x x x	Add mem word to addr with carry
SUB addr	0 2 x x x x x x	2 2 x x x x x x	$A \leftarrow A - \text{mem}$
SBB addr	0 3 x x x x x x	2 3 x x x x x x	sub mem word from addr with brow
AND addr	0 4 x x x x x x	2 4 x x x x x x	A .AND .addr
XOR addr	0 5 x x x x x x	2 5 x x x x x x	A .XOR .addr
OR addr	0 6 x x x x x x	2 6 x x x x x x	A .OR .addr
MAM addr	0 7 x x x x x x	2 7 x x x x x x	$A \rightarrow M$
MBM addr	0 8 x x x x x x	2 8 x x x x x x	$B \rightarrow M$
MCM addr	0 9 x x x x x x	2 9 x x x x x x	$C \rightarrow M$
MDM addr	0 A x x x x x x	2 A x x x x x x	$D \rightarrow M$
MMA addr	0 B x x x x x x	2 B x x x x x x	$M \rightarrow A$
MMB addr	0 C x x x x x x	2 C x x x x x x	$M \rightarrow B$
MMC addr	0 D x x x x x x	2 D x x x x x x	$M \rightarrow C$
MMD addr	0 E x x x x x x	2 E x x x x x x	$M \rightarrow D$
CALL addr	0 F x x x x x x	2 F x x x x x x	Call address
JMP addr	1 0 x x x x x x	3 0 x x x x x x	Jump address
JC addr	1 1 x x x x x x	3 1 x x x x x x	Jump if Carry
JZ addr	1 2 x x x x x x	3 2 x x x x x x	Jump if Zero
JPE addr	1 3 x x x x x x	3 3 x x x x x x	Jump party Even
JM addr	1 4 x x x x x x	3 4 x x x x x x	Jump party Minus
ISZ addr	1 5 x x x x x x	3 5 x x x x x x	Increment and skip if it is Zero
MUL addr	1 6 x x x x x x	3 6 x x x x x x	$DA \leftarrow A \times \text{mem}$
DIV addr	1 7 x x x x x x	3 7 x x x x x x	$DA / \text{mem} \quad Q=A, R=D$
	1 8 x x x x x x	3 8 x x x x x x	Future
	1 9 x x x x x x	3 9 x x x x x x	Future
	1 A x x x x x x	3 A x x x x x x	Future
	1 B x x x x x x	3 B x x x x x x	Future
	1 C x x x x x x	3 C x x x x x x	Future
	1 D x x x x x x	3 D x x x x x x	Future
	1 E x x x x x x	3 E x x x x x x	Future

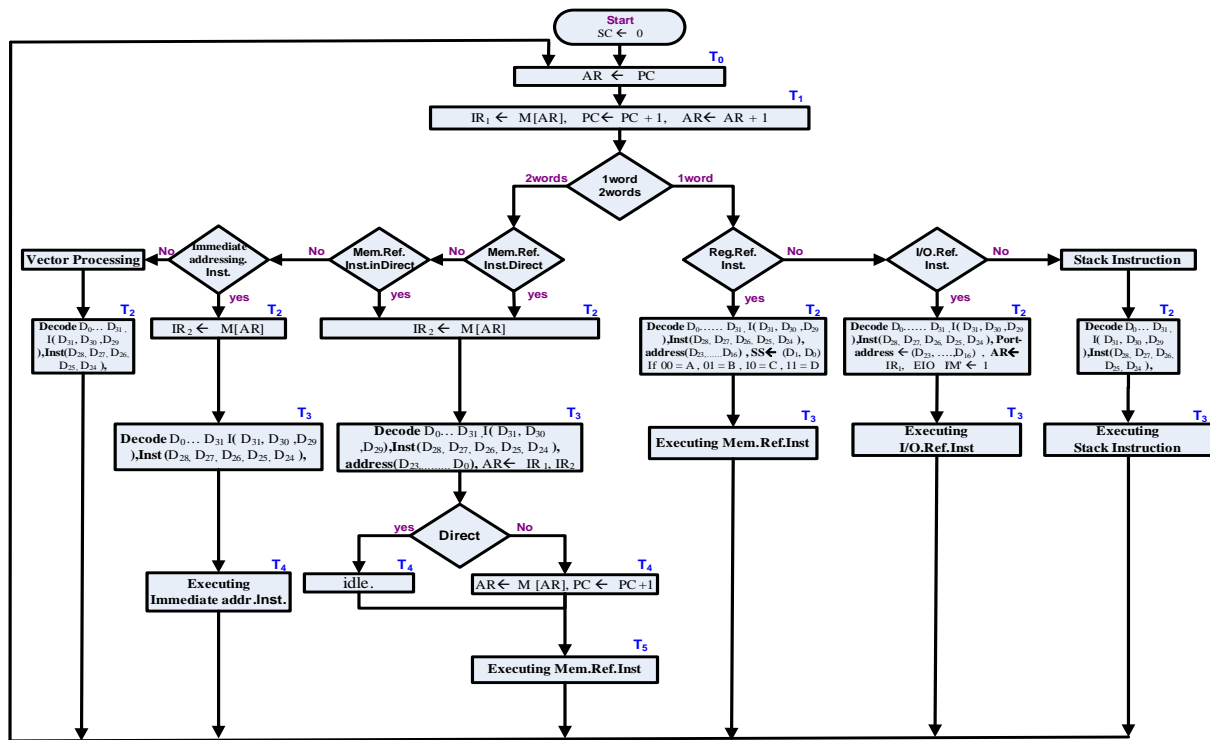


Fig.2. Instruction Cycle Flow of the A-MISC

The rest 8-bits of the first word denote the most significant 8-bits of the 24-bit address. The second word of the instruction code gives the least significant 16 bits of the address. The second code 001 for III denotes the memory reference instruction with indirect addressing feature. In organizing the indirect address the most significant 8-bits of the indirect address is kept same as the most significant 8-bits of the direct address. The least significant 16-bit word is different for direct address and the indirect address.

#### Register Reference Instructions

For a register reference instruction adding a register R ( A,B,C or D ) with A register with Carry flag is denoted by ADC R. In the instruction code of 16 bits the least significant two bits give the code of the register R used. The instruction category for register reference instructions for logical operations include AND R, OR R, XOR R and COM R. For arithmetic operations of register reference category we have CLR R, INC R, DEC R, ADD R, ADC R, SUB R, SBB R, CMP R, MUL R, and DIV R. Shifting and rotating instructions are SHL R, SHR R, ROR R, ROL R. Movement of data from one register to another comes under the category of MOV R,R. In the instruction code the least significant two bits give code for the source register and the next two bits denote the code of destination register.

Other register reference instructions are those affecting flags such as CLC (Clearing Carry), STC (setting Carry), CLZ (Clear Zero), STZ(Set Zero), CMC (Complement Carry), Skip instructions like SPA (Skip next instruction if A is positive), SNA (Skip next instruction if A is negative), SZA(Skip next instruction if A is zero), and those relating to decimal arithmetic DAA(Decimal adjust), CWD(Convert word to double

word) and HLT. The DAA instruction will be used for decimal arithmetic operations. Division operations could use CWD instruction.

#### I/O Reference Instructions and Others

The I/O reference instructions include INP and OUT instructions. While all the I/O reference instructions have the III code as 011, INP instruction has the 5 bits opcode as 00000 and the OUT instruction has the opcode 00001. The word length of the port address in either case is 8 bits only. These 8 bits are accommodated in the least significant 8 bits of the instruction word. The software interrupt INT instructions are included in this group and there are 8 such instructions available. The instructions affecting the interrupt flag namely STI setting the interrupt flag and CLI clearing it are parts of this group.

Stack instructions are regular PUSH and POP instructions. SP the Stack pointer of 24-bits word length always keep the top of the stack. While PUSH instruction decrements the SP by one step the POP instruction increments by one.

By immediate addressing instruction one could load the immediate data word into any register.

#### Instruction Cycle

The instruction cycle doing all operations undergoes fetch, decode and execute phases in sequence. Fetch and decode operations are first two phases in all types of instructions. After this, by decoding the III bits the instruction category are identified and the execution phase is carried out accordingly. Fig.2 shows the phases of operation performed in an instruction cycle. The second instruction word is handled accordingly.

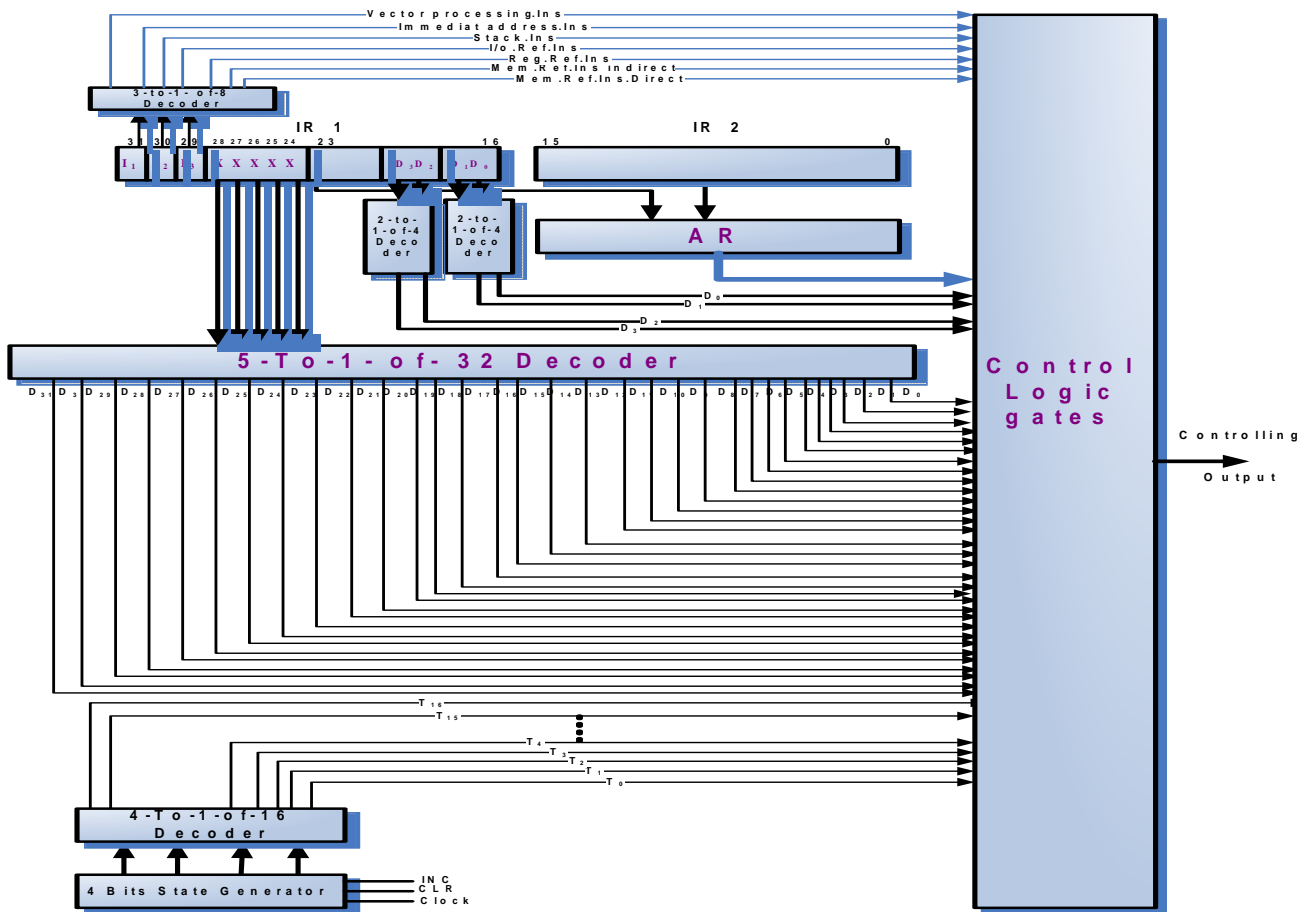


Fig.3. Simplified Schematic of the Control Unit

#### Completeness of Instruction set

The architecture of the A-MISC includes all necessary instructions required for a computer for solving general purpose problem. It has instructions performing unconditional jumps and conditional jumps. Conditional jumps are based on the flags set currently. Essential flags such as Carry, Zero, Parity even, Sign, auxiliary Carry, Interrupt, Trap and Overflow are available here.

#### Interrupts

We have software interrupts performed by the 8 numbers of INT instructions. The memory area for service procedure of these interrupts are reserved. It can go to a maximum of 256 words. The starting address of the procedure for INT n instruction is given by 000n00. Therefore INT 5 instruction has its starting address at Hex 000500 and the ending address would be Hex 0005FF.

#### Hardware Interrupts

Two hardware interrupts INTR1 and INTR2 are allowed in the system. When INTR1 occurs the system runs automatically the INT 0 instruction and when INTR2 occurs it runs INT 1 instruction. Therefore these two INT instructions are reserved for the hardware interrupts and not available for the user.

#### 4. CONTROL UNIT OF THE A-MISC

The control unit of the CPU is designed on hardwired concept and the hardware of the whole control unit has been developed. The two instruction registers IR1

and IR2 receive the instruction codes during instruction fetch phase and the decoders attached to that registers identify the micro operations to be carried out for entering into the next phase of operation. One word instructions just get the code shifted to IR1 register only. Fig.3 shows the simplified schematic of control unit designed on hardwired approach. All control logic gates performing various operations are designed and implemented.

The data is moved between the registers and the memory through common bus. Therefore transfer of data are performed in a shorter time. Fig.4 shows its simplified schematic of common bus. The selection codes for various registers are also indicated and designed accordingly. There are two parts of ALU designed for A-MISC. ALU x performs all arithmetic and logic operations except multiplication and division operation. Multiplication and division operations are done in ALU%. Whenever division instruction occurs it selects this ALU to divide the word at D and A register with DR register and put the quotient and remainder in A and D register respectively. The MUL instruction multiplies the contents of A register with that of the DR register and loads the 32-bit result in D and A registers.

A-MISC includes all important flags needed for a high performance CPU. Arithmetic and logic instructions affect most flags (Carry, Auxiliary Carry, Zero, Sign, parity, Overflow, Interrupt and Trap) and when such instructions occur the flag register is updated accordingly.

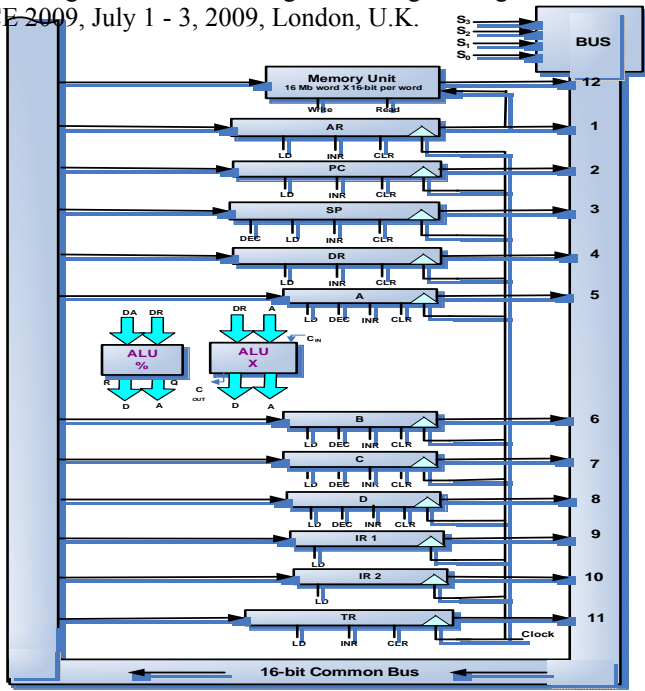


Fig.4. Common Bus and Selection Codes

register. Flags are changed by the user by respective instructions and also by PUSH flags and POP flags instructions.

5. ASSEMBLER FOR A-MISC

The assembler of the A-MISC has been developed in Borland C. It is an emulator and It has two passes. The symbolic assembly program written for A-MISC is first

given to the first pass as a data file. This generates the address symbol table saving the address symbols used in the program together with the location counter. For the sake of simplicity the first pass is not described elaborately. The second pass program having many tables as data files accepts the symbolic assembly program and the address symbol table constructed by pass1 as to construct the object file and save it in memory. The groups of bits required for constituting the instruction code are obtained from different resources such as tables and the instruction code is assembled. Fig.5 shows the sequence of operation done by second pass. For instance if the current assembly instruction is a register reference instruction, say MOV A,B, which moves the contents of A register to B register, the code for MOV, register codes A and B, III code and opcode are taken from the tables and the instruction code is constructed. For memory reference instruction it needs address for the address symbol used and it is taken from the address symbol table created by the first Pass. Also depending upon the presence of I in the assembly instruction III code is taken and used. In order to distinguish the address symbol from the immediate data and the register symbols, the address symbol used in the A-MISC should start from the letter G onwards. The location counter LC responsible for generating the address for the object code is incremented after assembling each instruction. Syntax error, if any, present in the assembly instruction is identified and indicated accordingly.

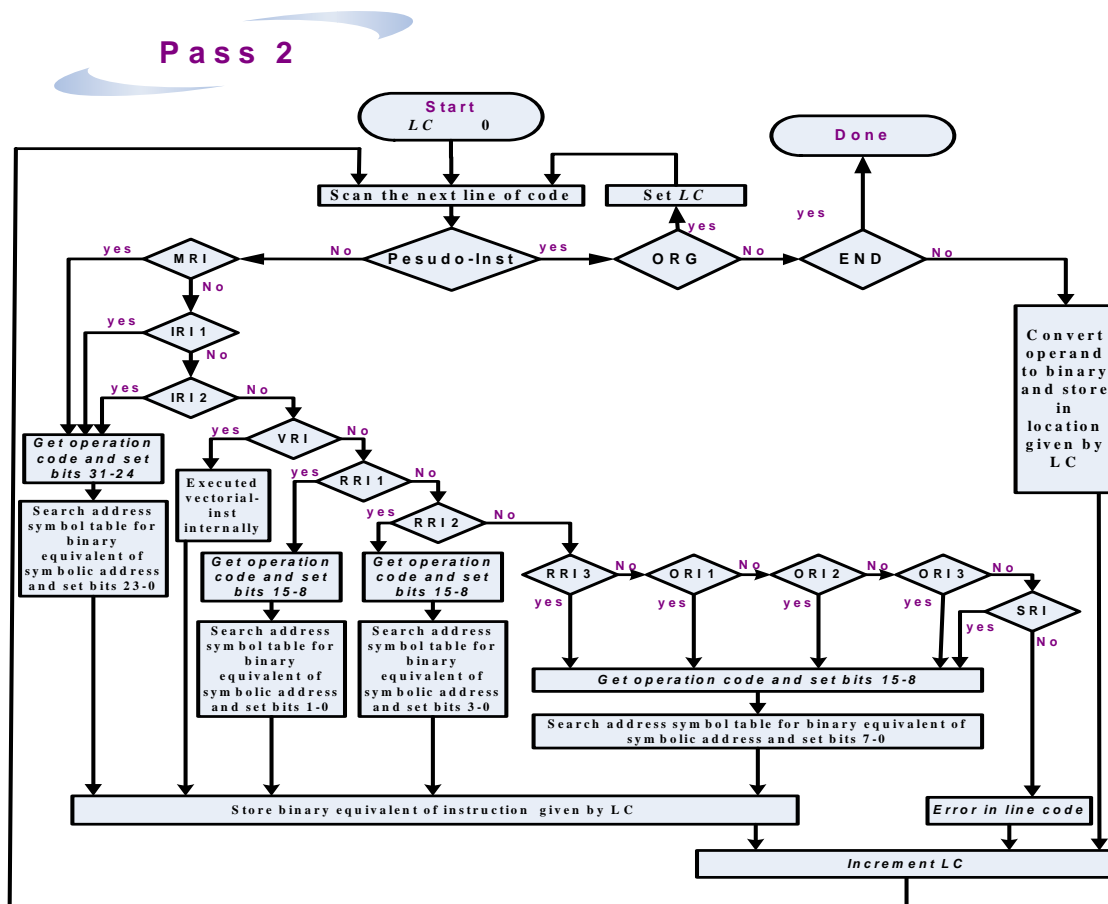


Fig.5. Sequence of Operations in Pass2

Also an emulator platform is developed for A-MISC to enter assembly program in PC which uses the A-MISC assembler to develop object program and run it in PC. In the emulator package it opens window for entering the assembly instructions for A-MISC, call the assembler, execute the instructions and see the results in memory and registers. Other windows appearing in the menu are for registers, flags and memory. In this package the assembly programs saved in discs could also be uploaded and assembled accordingly.

## 6. RESULTS AND DISCUSSION

Owing to the space limitations, the report for the project is presented concisely giving information very briefly. The complete hardware for the computer has been developed with hardwired control operation. The design of logic gates controlling various registers, memory, flip flops and common bus are supported with the design chart (not discussed here) conceding the control expressions and the operations performed. As the next phase of the development for the processor, vector processing instructions are being included in the instruction set and is being conceived in the hardware. A co-processor attached with 64K memory of its own would take care of vector processing operations. Most popular vector instructions such as vector addition, matrix inversion and matrix multiplication are proposed to be included. Whenever vector instruction is encountered the A-MISC would transfer the instruction to the coprocessor. Also, the vector table from main memory is transferred into the local memory of the coprocessor through DMA.. After transferring the data the coprocessor does its job independently and when completed it would inform the A-MISC by interrupt. Subsequently, the processed data stream is transferred to main memory of the A-MISC by DMA.

Peripherals like keyboard and printer are provided with appropriate interfaces extended to the CPU through the I/O ports. Since the port address is of 8-bits word length we can design up to 256 I/O ports and this would meet most peripheral hardware needs. As the design of ports are well known they are not explained here. For extended hardware peripheral interface requirements, if further ports are required then it could be realized using memory mapped I/O configuration.

With the extensive use of instructions in the instruction set several programs processing the arrays and also the procedures for different purposes have been developed, assembled and run with A-MISC. For several software requirements direct instructions are mostly available in the instruction set and wherever not found they are realized indirectly by using more than one instructions. For instance multi-precision arithmetic, although not available as exclusive instructions in the instruction set, it is achieved indirectly by using few data words for each operand and use the appropriate instructions to meet the needed arithmetic operation. For a standard application program the object program size is compared with that of the other processors and found that it is comparable and simpler in some instances. The processing time is evaluated in terms of clock cycles and again found lower compared to that of several

processors. In some special instances such as multi-precision arithmetic, there may not be much saving in time compared to CISC computers due to the usage of multiple instructions for the same task. A simulation program has been developed to estimate the correctness, completeness and efficiency of the hardware employed in the A-MISC. The minor errors encountered in the hardware design were detected by the simulation program and have been corrected accordingly. The clock frequency to be applied to the CPU depends on the electronic technology used and if the CPU chip is developed according to the latest technology available it is feasible to use the frequency as 2.5GHz.

## 7. CONCLUSION

The instruction code has only two possible sizes of 16 bits accommodated in one word or 32-bits realized with two 16 bit words. Compared to most CISCs this has a simplified hardware configuration. Furthermore, standard word length of 16 bits for the processor data makes the hardware processing aspects simplified further. As the space provided for the opcode in the instruction codes are only partially utilized the instruction set has scope for expansion with inclusion of additional possible instructions in different categories. The compilers for few high level languages are being worked out and hope to be completed in the near future.

The design principle put forth in this paper could easily be extended to conceive large word size processors such as 32-bit processors with address word size extending more than 32 bits. Several modeling and simulation programs perform a great deal analytical tasks and the A-MISC instruction set affords to accommodate most such deals.

## ACKNOWLEDGMENT

The author acknowledges with thanks all necessary support extended by the Department.

## REFERENCES

- [1] Morris Mano, Computer System Architecture, PHI (2004).
- [2] William Stallings, Computer Organization and Architecture, PHI (2002).
- [3] Rafiqzaman, Microprocessor and Microcomputer based design, Univ Book Stall (2004).
- [4] D.Hall, Microprocessors and Interfacing 8086, 80386, 80486, McGraw Hill (2004).
- [5] B.Bray, The Intel Microprocessors 8086, 80186, 80286, 80386 and 80486 and Pentium, Prentice Hall (2003).
- [6] [http://en.wikipedia.org/wiki/Complex\\_instruction\\_set\\_computer](http://en.wikipedia.org/wiki/Complex_instruction_set_computer) (May 2006).
- [7] <http://acorn.riscos.com/> (May 2006).
- [8] <http://en.wikipedia.org/wiki/RISC> (May 2006).
- [9] <http://www.risc-inc.com/> (May 2006).
- [10] <http://www.geocities.com/SiliconValley/Chip/5014/> (May 2006).
- [11] <http://www.arm.com/> (May 2006).
- [12] RISC Processor, The Columbia Encyclopedia, Sixth Edition (2001-05).
- [13]