Micro-Baby

 Micro-Baby is a simple computer architecture, in fact, very simple. All microcontrollers and microprocessors are computer architectures, in most cases fairly simple ones. However, in today’s world even microcontrollers and microprocessors are eons beyond basic. They include many advanced features which often obscure the basic concepts of a stored program computer.

 It is assumed that the reader of this book and this chapter possesses a basic understanding of the binary number system and the implementation of logic equations in digital logic using AND, OR, NAND, NOR, and NOT gates. If you wish to brush up a quick review of these topics is provided in Appendix A and Appendix B.

 Micro-Baby is an accumulator based load-store architecture. In fact, that is a good place to start this chapter. The basic concepts of a stored program computer will be presented and how it can be implemented with this architecture. Just like the architecture of a home, the architecture of a computer is the ‘how’ the pieces are put together. Building on this concept the instructions of Micro-Baby will be presented, the stored program that allows Micro-Baby to do something useful.

A. Accumulator Based Load-Store Architecture



The Micro Baby implementation of this has the following structure:



MircoBaby instructions:

Instructions are 2 bytes in length. The first byte is the op-code and the second where in data memory the 2nd operand is located. The first operand is in the accumulator and that is where the result will also be.

Arithmetic: (0100 0xxx)

ADD Add value to accumulator

ADDC Add value and carry to accumulator

SUB Subtract value from accumulator

SUBC Subtract value and borrow from accumulator

INC Increment accumulator

DEC Decrement accumulator

Logical (0101 xxxx)

AND Logically AND the accumulator and value

OR Logically OR the accumulator and value

INV Logically invert the accumulator

XOR Logically XOR the accumulator and value

CLRA Clear Accumulator

SLA *not sure yet*

SRA *not sure yet*

Jumps: (64) (starts 11xx xyyy) – bit x indicates used or not, y indicates clear or set

Carry clear or set

Zero clear or set

Negative clear or set

Jump always – when not using any of the conditions (11000xxx)

Tests (0100 1xxx)

CMP Compare (executes Accumulator – argument) does not modify accumulator.

Data Movement (10y x xxmm) - 2 byte instruction – 2nd byte is address

 mm is addressing mode

LDA Load Accumulator (101x xxxmm)

STA Store Accumulator (100x xxxmm)

ADDRESSING MODES

Immediate

Direct

Indirect *Not sure yet*

**Memory**

Data divided into data memory and instruction memory

Each is 256 bytes – addressable by 8 bits

CPU to memory signals – 8-bit data bus, 8-bit address bus, r/w signal, and timing with is used to generate and enable signal. Data is placed on the bus when enable is valid. Data, address, and r/w go to high impedance when not needed by bus master.

**mb Controller**

The controller contains the Program counter, the instruction register, and the status register.

The main part of the controller is a state machine which alternates between two main states, Fetch and Execute. Within each of these there are multiple subcycles.

The first cycle completes the action MEM(PC) 🡪 IR

 PC + 1 🡪 PC

The second cycle then performs the action of the instruction generating the control signals for the datapath to execute the instruction.

**HDL model**

The memory module is specified as follows:

Each memory block (data or instruction) has the following features as indicated by the diagram.

 

The bus timing looks as follows: (the control signals need to be added.



The ALU has the following structure and control signals.



The datapath has the following structure and control signals



The controller has the following structure.



**Assignment VP2:**

1. Verify the Microbaby structures available. The one structure that is not yet complete is the controller. Work on completing its design and verification after completing verification of the components of the architecture.

2. Write a testplan to test and verify the components of the architecture. Testplan is due the Monday December 1st.

3. Work on design of the controller. A microcoded controller is desired. There is high probability a behavioral controller will be available from the design group soon.

4. Write a final verification report that details your outcomes on these tasks.