**Programming Project on Simulator for APEX-Part I**

**Kunal Zantye**

**Email :** [**kzantye1@binghamton.edu**](mailto:kzantye1@binghamton.edu)

**Date :11/18/2015**

This part of the project implements a cycle-by-cycle simulator for the simple 5-stage APEX pipeline. It is assumed that the code to be simulated is stored in a text file with one ascii string representing an instruction (in the symbolic form, such as **ADD R1 R4 R6**) in each line of the file.

The instructions supported are:

* Register-to-register instructions: ADD, SUB, MOVC, MUL, AND, OR, EX-OR (all done on the ALU in a single cycle). You can assume that the result of multiplying two registers will fi into a single register.
* Memory instructions: LOAD, STORE
* Control flow instructions: BZ, BNZ, JUMP, BAL, HALT

It is assumed that the register to register instruction do not contain literals

As given there are 8 architectural registers, R0 through R7. Memory is viewed as a linear array of integer values (4 Bytes wide) numbered 0 through 9999.

The BAL register implements function calls, saves the return address in a special register X and transfers control to a specified target address:

BAL <register>, literal: saves address of next instruction in X and then sets fetch PC to contents of <reg> plus the literal.

MOVC <register> <literal>, moves literal value into specified register

The HALT instruction stops execution. JUMP specifies a register and a literal and transfers control to the address obtained by adding the contents of the register to the literal.

To invoke the simulator

1. Run the executable file
2. Write the name of the input file that contains the code to be simulated. (For example: input.txt)

The simulator have a command interface that allows users to execute the following commands:

1. Initialize: Initializes the simulator state, sets the PC of the fetch stage to point to the first instruction in the ascii code file, which is assumed to be at address 20000. Each instruction takes 4 bytes of space, so the next instruction is at address 20001, as memory words are 4 Bytes long, just like the integer data items.
2. Simulate <n>: simulates the number of cycles specified as <n> and waits. Simulation can stop earlier if a HALT instruction is encountered and when the HALT instruction is in the WB stage.
3. Display: Displays the contents of each stage in the pipeline and the contents of the first 100 memory locations containing data, starting with address 0.

**Design Implementation**

The gant chart representation is shown below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Cycle 1 | Cycle2 | Cycle 3 | Cycle 4 | Cycle 5 | Cycle 6 | Cycle 7 | Cycle 8 | Cycle 9 |
| Fetch | I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | I9 |
| Decode |  | I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 |
| Exe |  |  | I1 | I2 | I3 | I4 | I5 | I6 | I7 |
| Memory |  |  |  | I1 | I2 | I3 | I4 | I5 | I6 |
| WB |  |  |  |  | I1 | I2 | I3 | I4 | I5 |

Where Fetch, Decode, Exe, Memory and Writeback are five stages of Apex pipeline. I1,I2… are is the number of instruction.

In the case of dependency the instruction are stalled in the decode stage. When the previous instruction passes to the writeback stage, the processes is resumed.

**Code Implementation**

The simulator is written in C language.

There are five function to perform the different stages of the apex pipelining. Below is the description of these function:

1. Fetch()

It reads the single line from the input file which contains the instruction.

For example: Instruction 49 Fetched: 20049 STORE R7 R2 78

1. Decode()

It decodes the instruction fetched by the fetch function in the previous cycle.

It identifies the type of instruction , that is whether it is a register- register instruction or LOAD or STORE instruction or control instruction.

It also decodes the value of the register from register file and the destination register address

For example: If the fetched instruction is ADD R5 R5 R0

Decoded Instruction: ADD R5 42 22

1. Exe()

It acts as the ALU.

It performs the calculation like ADD, SUB, MUL, OR, AND and EX-OR

For example: If the decode instruction is ADD R5 42 22

Execution result: 64

1. Memory()

The instruction other than LOAD or STORE is only passed through this stage using latch.

In case of LOAD instruction, it fetches the value from memory location calculated by exe() function.

For example :

In case of register to register instruction:

Fetched instruction: SUB R4 R6 R0

Result from Exe function: 101

Mem Stage(pasing result using latch): R4 101

In case of STORE instruction, it stores the value to the memory location calculated by exe() function.

1. WB()

Updates the register files.

It basically writes the result to the resister file

Below is the example of the register file contains:

WriteBack Stage: ||R0:22 ||R1:1464 ||R2:0 ||R3:3906 ||R4:93 ||R5:42 ||R6:123 ||R7:22 ||

**Example Results**

**For test input 1:**

1]Test result for clock cycles=50:

*Kunals-MBP:~ KUNAL$ ./apex*

*Enter the input text file: input1.txt*

*Enter the command ($initialize,$simulate n,$display)*

*initialize*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*simulate 50*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*display*

*completed*

*Memory*

*M0 : 0 M1 : 0 M2 : 0 M3 : 0 M4 : 0 M5 : 0 M6 : 0 M7 : 0 M8 : 0 M9 : 0 M10 : 0 M11 : 0 M12 : 0 M13 : 5 M14 : 0 M15 : 0 M16 : 0 M17 : 0 M18 : 0 M19 : 0 M20 : 0 M21 : 0 M22 : 0 M23 : 0 M24 : 0 M25 : 0 M26 : 0 M27 : 0 M28 : 0 M29 : 0 M30 : 0 M31 : 0 M32 : 0 M33 : 0 M34 : 0 M35 : 0 M36 : 0 M37 : 0 M38 : 0 M39 : 0 M40 : 0 M41 : 0 M42 : 0 M43 : 0 M44 : 0 M45 : 0 M46 : 0 M47 : 0 M48 : 0 M49 : 0 M50 : 0 M51 : 0 M52 : 0 M53 : 0 M54 : 0 M55 : 0 M56 : 0 M57 : 0 M58 : 0 M59 : 0 M60 : 0 M61 : 0 M62 : 0 M63 : 0 M64 : 0 M65 : 0 M66 : 0 M67 : 0 M68 : 0 M69 : 0 M70 : 0 M71 : 0 M72 : 0 M73 : 0 M74 : 0 M75 : 0 M76 : 0 M77 : 0 M78 : 30 M79 : 0 M80 : 0 M81 : 0 M82 : 0 M83 : 0 M84 : 0 M85 : 0 M86 : 0 M87 : 0 M88 : 0 M89 : 0 M90 : 0 M91 : 0 M92 : 0 M93 : 0 M94 : 0 M95 : 0 M96 : 0 M97 : 0 M98 : 0 M99 : 0*

*Instruction 49 Fetched: 20049 STORE R7 R2 78*

*Decoded Instruction: ADD R5 42 22*

*Execution result: 56*

*Mem Stage(pasing result using latch): R4 101*

*WriteBack Stage: ||R0:22 ||R1:1464 ||R2:0 ||R3:3906 ||R4:93 ||R5:42 ||R6:123 ||R7:22 ||*

2]Test result for clock cycles=40+40=80

*Kunals-MBP:~ KUNAL$ ./apex*

*Enter the input text file: input1.txt*

*Enter the command ($initialize,$simulate n,$display)*

*initialize*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*simulate 40*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*simulate 40*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*display*

*completed*

*Memory*

*M0 : 0 M1 : 0 M2 : 0 M3 : 0 M4 : 0 M5 : 0 M6 : 0 M7 : 0 M8 : 0 M9 : 0 M10 : 0 M11 : 0 M12 : 0 M13 : 5 M14 : 0 M15 : 0 M16 : 0 M17 : 0 M18 : 0 M19 : 0 M20 : 0 M21 : 0 M22 : 0 M23 : 0 M24 : 0 M25 : 0 M26 : 0 M27 : 0 M28 : 0 M29 : 0 M30 : 0 M31 : 0 M32 : 0 M33 : 0 M34 : 0 M35 : 0 M36 : 0 M37 : 0 M38 : 0 M39 : 0 M40 : 0 M41 : 0 M42 : 0 M43 : 0 M44 : 0 M45 : 0 M46 : 0 M47 : 0 M48 : 0 M49 : 0 M50 : 0 M51 : 0 M52 : 0 M53 : 0 M54 : 0 M55 : 0 M56 : 0 M57 : 0 M58 : 0 M59 : 0 M60 : 0 M61 : 0 M62 : 0 M63 : 0 M64 : 0 M65 : 0 M66 : 0 M67 : 0 M68 : 0 M69 : 0 M70 : 0 M71 : 0 M72 : 0 M73 : 0 M74 : 0 M75 : 0 M76 : 0 M77 : 0 M78 : 3906 M79 : 0 M80 : 0 M81 : 0 M82 : 0 M83 : 0 M84 : 0 M85 : 0 M86 : 0 M87 : 0 M88 : 0 M89 : 0 M90 : 0 M91 : 0 M92 : 0 M93 : 0 M94 : 0 M95 : 0 M96 : 0 M97 : 0 M98 : 0 M99 : 0*

*Instruction 79 Fetched: 20079 SUB R4 R6 R0*

*Decoded Instruction: STORE -3783 5434 123*

*Execution result: -20556822*

*Mem Stage: R2 0*

*WriteBack Stage: ||R0:6464 ||R1:-2049048 ||R2:0 ||R3:-2049048 ||R4:-3783 ||R5:5434 ||R6:123 ||R7:6464 ||*

**For test input 2:**

1]Test result for clock cycles=20

*Kunals-MBP:~ KUNAL$ ./apex*

*Enter the input text file: input1.txt*

*Enter the command ($initialize,$simulate n,$display)*

*initialize*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*simulate 20*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*display*

*completed*

*Memory*

*M0 : 0 M1 : 0 M2 : 0 M3 : 0 M4 : 0 M5 : 0 M6 : 0 M7 : 0 M8 : 0 M9 : 0 M10 : 0 M11 : 0 M12 : 0 M13 : 0 M14 : 0 M15 : 0 M16 : 0 M17 : 0 M18 : 0 M19 : 0 M20 : 0 M21 : 0 M22 : 0 M23 : 0 M24 : 5 M25 : 0 M26 : 0 M27 : 0 M28 : 0 M29 : 0 M30 : 0 M31 : 0 M32 : 0 M33 : 0 M34 : 0 M35 : 0 M36 : 0 M37 : 0 M38 : 0 M39 : 0 M40 : 0 M41 : 0 M42 : 0 M43 : 0 M44 : 0 M45 : 0 M46 : 0 M47 : 0 M48 : 0 M49 : 0 M50 : 0 M51 : 0 M52 : 0 M53 : 0 M54 : 0 M55 : 0 M56 : 0 M57 : 0 M58 : 0 M59 : 0 M60 : 0 M61 : 0 M62 : 0 M63 : 0 M64 : 0 M65 : 0 M66 : 0 M67 : 0 M68 : 0 M69 : 0 M70 : 0 M71 : 0 M72 : 0 M73 : 0 M74 : 0 M75 : 0 M76 : 0 M77 : 0 M78 : 0 M79 : 0 M80 : 0 M81 : 0 M82 : 0 M83 : 0 M84 : 0 M85 : 0 M86 : 0 M87 : 0 M88 : 0 M89 : 0 M90 : 0 M91 : 0 M92 : 0 M93 : 0 M94 : 0 M95 : 0 M96 : 0 M97 : 0 M98 : 0 M99 : 0*

*Instruction 19 Fetched: 20019 MOVC R1 2 \**

*Decoded Instruction: OR R1 0 5*

*Execution result: 3*

*Mem Stage: R2 0*

*WriteBack Stage: ||R0:7 ||R1:0 ||R2:0 ||R3:4 ||R4:-3 ||R5:6 ||R6:7 ||R7:8 ||*

2]Test result for clock cycles=40

*Kunals-MBP:~ KUNAL$ ./apex*

*Enter the input text file: input1.txt*

*Enter the command ($initialize,$simulate n,$display)*

*initialize*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*simulate 40*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*display*

*completed*

*Memory*

*M0 : 0 M1 : 0 M2 : 0 M3 : 0 M4 : 0 M5 : 0 M6 : 0 M7 : 0 M8 : 0 M9 : 0 M10 : 6 M11 : 0 M12 : 0 M13 : 0 M14 : 0 M15 : 0 M16 : 0 M17 : 0 M18 : 0 M19 : 0 M20 : 0 M21 : 0 M22 : 0 M23 : 0 M24 : 0 M25 : 0 M26 : 0 M27 : 0 M28 : 0 M29 : 0 M30 : 0 M31 : 0 M32 : 0 M33 : 0 M34 : 0 M35 : 0 M36 : 0 M37 : 0 M38 : 0 M39 : 0 M40 : 0 M41 : 0 M42 : 0 M43 : 0 M44 : 0 M45 : 0 M46 : 0 M47 : 0 M48 : 0 M49 : 0 M50 : 0 M51 : 0 M52 : 0 M53 : 0 M54 : 0 M55 : 0 M56 : 0 M57 : 0 M58 : 0 M59 : 0 M60 : 0 M61 : 0 M62 : 0 M63 : 0 M64 : 0 M65 : 0 M66 : 0 M67 : 0 M68 : 0 M69 : 0 M70 : 0 M71 : 0 M72 : 0 M73 : 0 M74 : 0 M75 : 0 M76 : 0 M77 : 0 M78 : 0 M79 : 0 M80 : 0 M81 : 0 M82 : 0 M83 : 0 M84 : 0 M85 : 0 M86 : 0 M87 : 0 M88 : 0 M89 : 0 M90 : 0 M91 : 0 M92 : 0 M93 : 0 M94 : 0 M95 : 0 M96 : 0 M97 : 0 M98 : 0 M99 : 0*

*Instruction 39 Fetched: 20039 MOV R2 R1 \**

*Decoded Instruction: MOVC R1 2 0*

*Execution result: 0*

*Mem Stage(pasing result using latch): R2 -2*

*WriteBack Stage: ||R0:2 ||R1:0 ||R2:0 ||R3:4 ||R4:0 ||R5:0 ||R6:4 ||R7:8 ||*

**For test input 3:**

1]Test result for clock cycles=100

*Kunals-MBP:~ KUNAL$ ./apex*

*Enter the input text file: input1.txt*

*Enter the command ($initialize,$simulate n,$display)*

*initialize*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*simulate 100*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*display*

*completed*

*Memory*

*M0 : 0 M1 : 0 M2 : 0 M3 : 0 M4 : 0 M5 : 0 M6 : 0 M7 : 0 M8 : 0 M9 : 0 M10 : 0 M11 : 0 M12 : 0 M13 : 0 M14 : 0 M15 : 0 M16 : 0 M17 : 0 M18 : 0 M19 : 0 M20 : 0 M21 : 0 M22 : 0 M23 : 0 M24 : 0 M25 : 0 M26 : 0 M27 : 0 M28 : 0 M29 : 0 M30 : 0 M31 : 0 M32 : 0 M33 : 0 M34 : 0 M35 : 0 M36 : 0 M37 : 0 M38 : 0 M39 : 0 M40 : 0 M41 : 0 M42 : 0 M43 : 0 M44 : 0 M45 : 0 M46 : 0 M47 : 0 M48 : 0 M49 : 0 M50 : 0 M51 : 0 M52 : 0 M53 : 0 M54 : 0 M55 : 0 M56 : 0 M57 : 0 M58 : 0 M59 : 0 M60 : 0 M61 : 0 M62 : 0 M63 : 0 M64 : 0 M65 : 0 M66 : 0 M67 : 0 M68 : 0 M69 : 0 M70 : 0 M71 : 0 M72 : 0 M73 : 0 M74 : 0 M75 : 0 M76 : 0 M77 : 0 M78 : 0 M79 : 0 M80 : 0 M81 : 0 M82 : 0 M83 : 0 M84 : 0 M85 : 0 M86 : 0 M87 : 0 M88 : 0 M89 : 0 M90 : 0 M91 : 0 M92 : 0 M93 : 0 M94 : 0 M95 : 0 M96 : 0 M97 : 0 M98 : 0 M99 : 0*

*Instruction 21 Fetched: SUB R1 R0 R2 20022*

*Execution result(JUMP): 9*

*Mem Stage(pasing result using latch): R2 15*

*WriteBack Stage: ||R0:93 ||R1:1 ||R2:15 ||R3:0 ||R4:0 ||R5:1 ||R6:0 ||R7:9 ||*

1]Test result for clock cycles=150

*Kunals-MBP:~ KUNAL$ ./apex*

*Enter the input text file: input1.txt*

*Enter the command ($initialize,$simulate n,$display)*

*initialize*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*simulate 150*

*completed*

*Do you want to continue(Y/N)?y*

*Enter the command ($initialize,$simulate n,$display)*

*display*

*completed*

*Memory*

*M0 : 0 M1 : 0 M2 : 0 M3 : 0 M4 : 0 M5 : 0 M6 : 0 M7 : 0 M8 : 0 M9 : 0 M10 : 0 M11 : 0 M12 : 0 M13 : 0 M14 : 0 M15 : 0 M16 : 0 M17 : 0 M18 : 0 M19 : 0 M20 : 0 M21 : 0 M22 : 0 M23 : 0 M24 : 0 M25 : 0 M26 : 0 M27 : 0 M28 : 0 M29 : 0 M30 : 0 M31 : 0 M32 : 0 M33 : 0 M34 : 0 M35 : 0 M36 : 0 M37 : 0 M38 : 0 M39 : 0 M40 : 0 M41 : 0 M42 : 0 M43 : 0 M44 : 0 M45 : 0 M46 : 0 M47 : 0 M48 : 0 M49 : 0 M50 : 0 M51 : 0 M52 : 0 M53 : 0 M54 : 0 M55 : 0 M56 : 0 M57 : 0 M58 : 0 M59 : 0 M60 : 0 M61 : 0 M62 : 0 M63 : 0 M64 : 0 M65 : 0 M66 : 0 M67 : 0 M68 : 0 M69 : 0 M70 : 0 M71 : 0 M72 : 0 M73 : 0 M74 : 0 M75 : 0 M76 : 0 M77 : 0 M78 : 0 M79 : 0 M80 : 0 M81 : 0 M82 : 0 M83 : 0 M84 : 0 M85 : 0 M86 : 0 M87 : 0 M88 : 0 M89 : 0 M90 : 0 M91 : 0 M92 : 0 M93 : 0 M94 : 0 M95 : 0 M96 : 0 M97 : 0 M98 : 0 M99 : 0*

*Instruction 19 Fetched: 20019 JUMP R1 20008 20020*

*Decoded Instruction: ADD R2 23 0*

*Execution result: 1*

*Mem Stage(pasing result using latch): 0*

*WriteBack Stage: ||R0:89 ||R1:1 ||R2:23 ||R3:0 ||R4:0 ||R5:0 ||R6:0 ||R7:9 ||*