NYU-EL 6463

ADVANCED COMPUTER HARDWARE DESIGN

Processor Design

&

RC5 Implementation

Chi Zhang cz1195

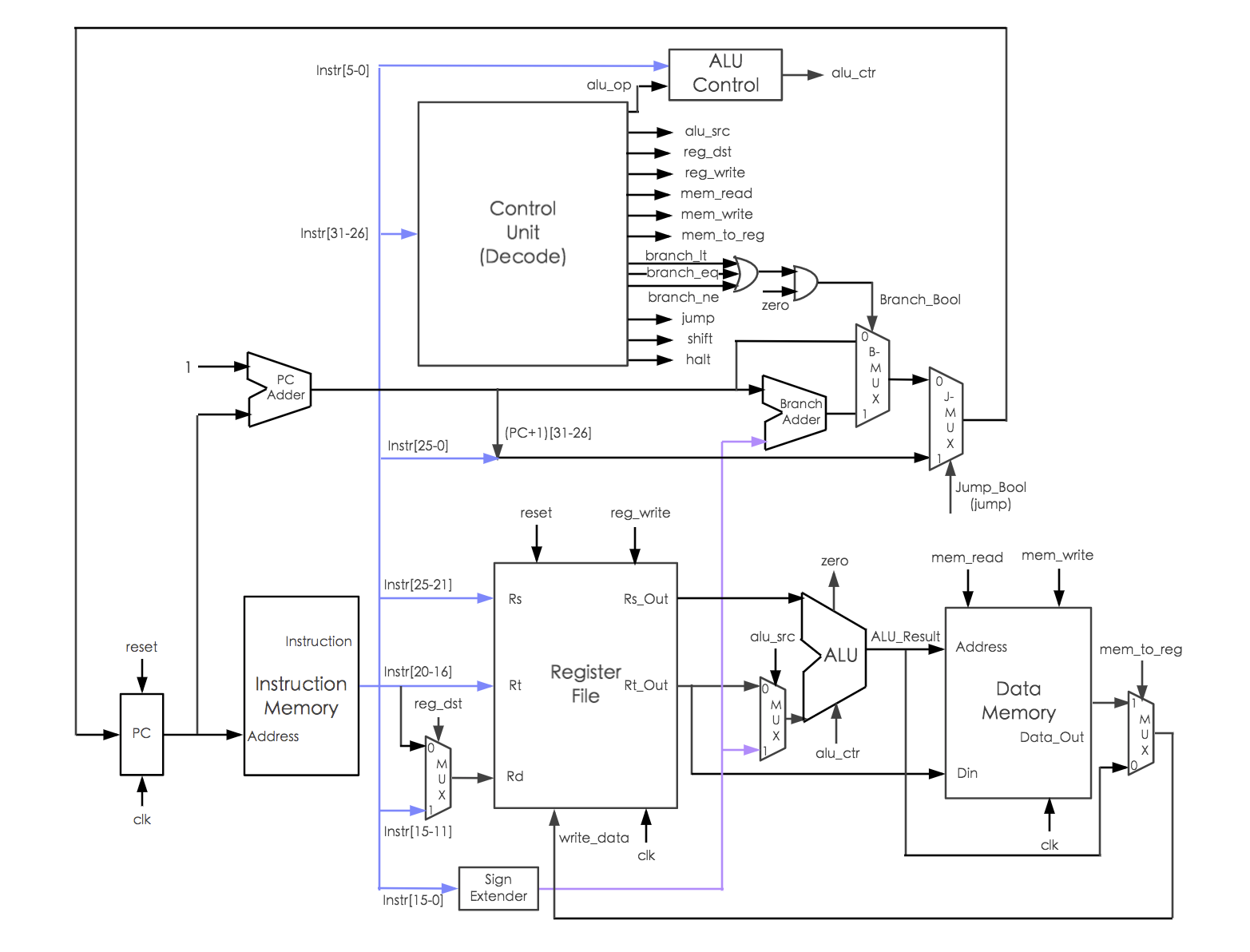
Hao Dong hd864

Jian Song js8612

Wen Dai wd580

Zijun Ma zm646

**Design block diagram**

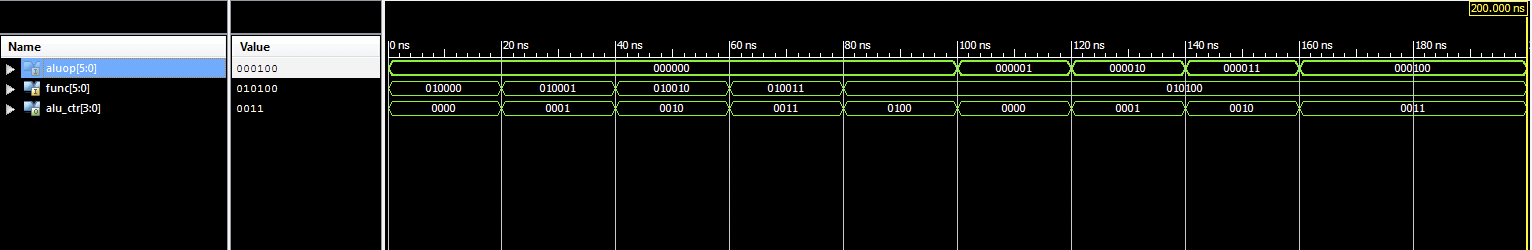


**ALU Design**

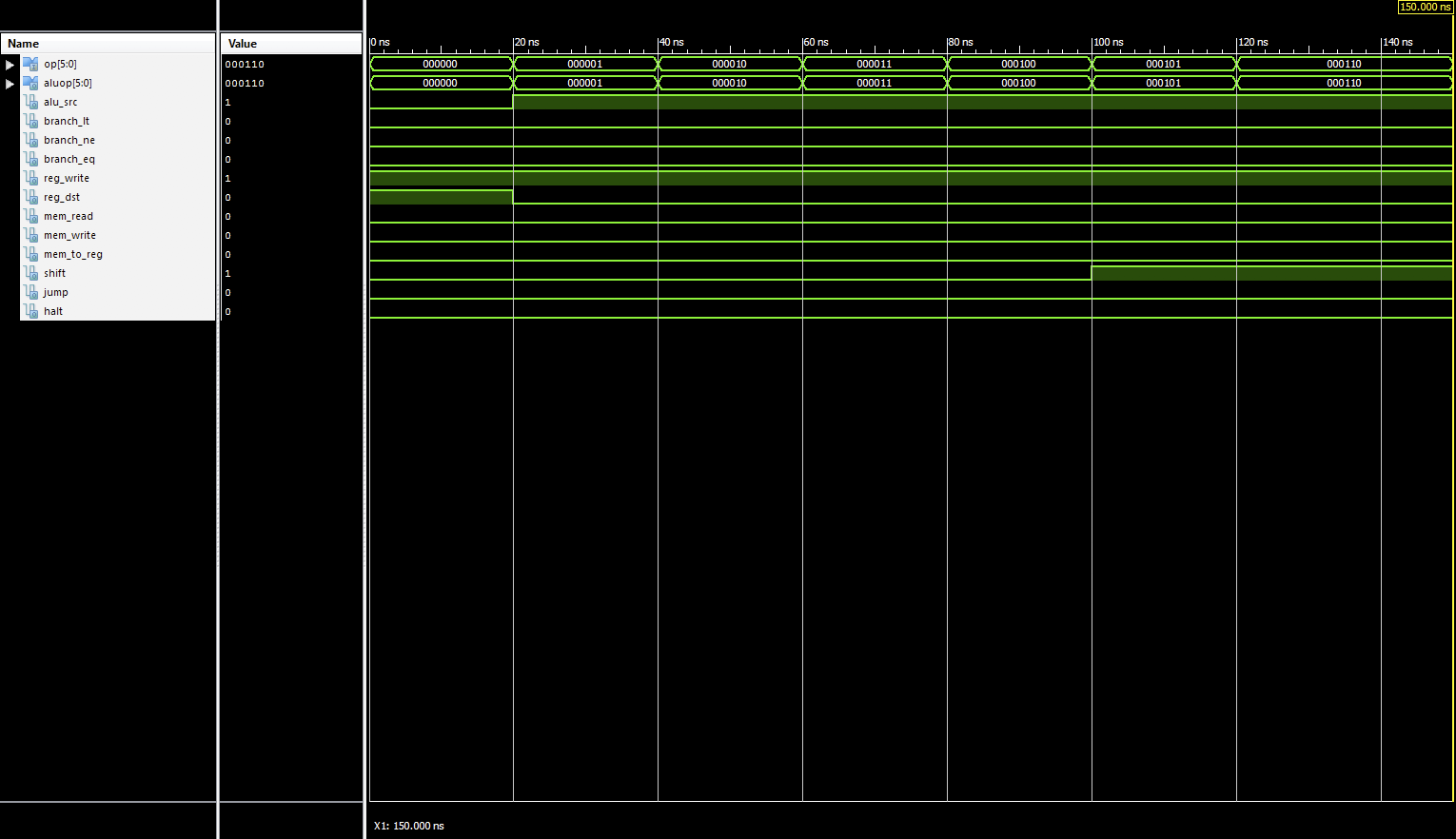


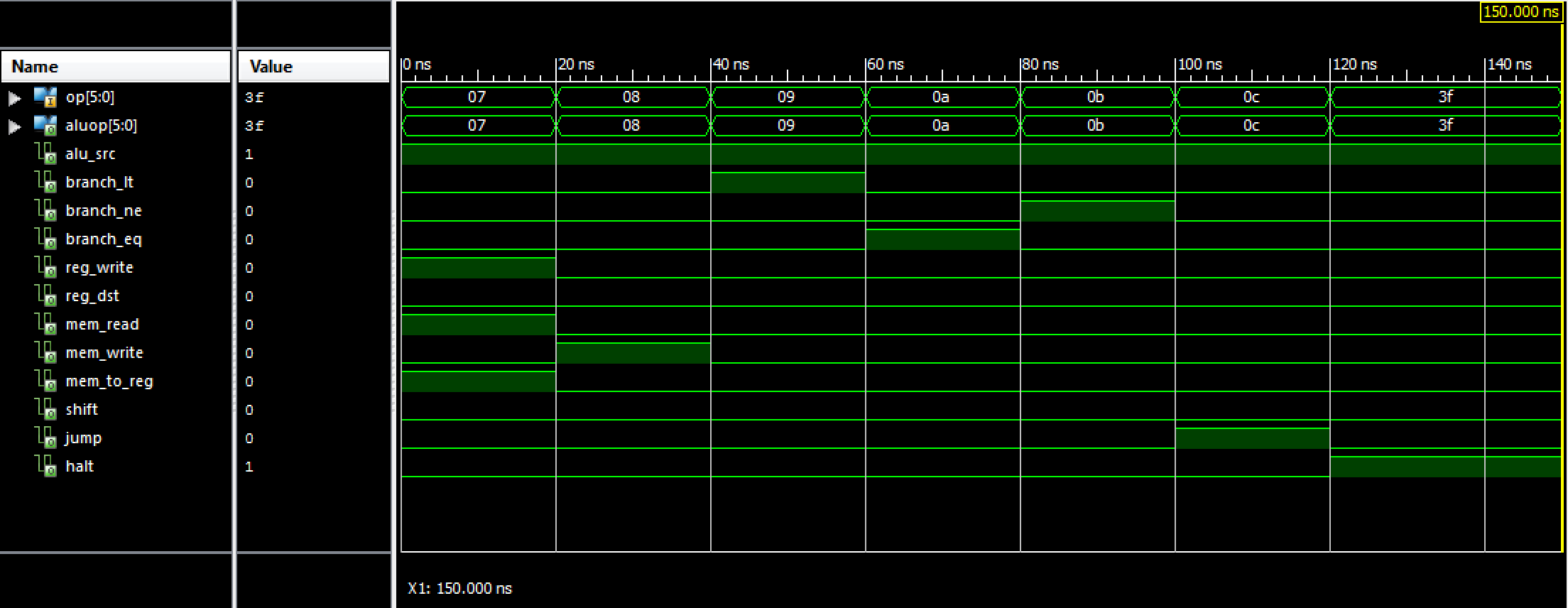
**Control Unit Design**

ALU Control



Instruction Decode





**Complete Processor Design**

Register File

Instruction Memory

Data Memory

**CPU Verification with Sample Programs**

Sample Program 1:

ADDI R1, R0, 7 // R1 = 7

ADDI R2, R0, 8 // R2 = 8

ADD R3, R1, R2 // R3 = R1 + R2 =15

HAL // HALT



Sample Program 2:

This program checks complete instruction set of 18 Instructions.

ADDI R1, R0, 2 // R0=0, R1=2

ADDI R3, R0, 10 // R3=10D (D=Decimal)

ADDI R4, R0, 14 // R4=14D

ADDI R5, R0, 2 // R5=2

SW R4, 2(R3) // 14D is stored in Data memory location 12D

SW R3, 1(R3) // 10D is stored in Data memory location 11D

SUB R4, R4, R3 // R4=2

SUBI R4, R0, 1 // R4=-1D

AND R4, R2, R3 // R4=0

ANDI R4, R2, 10 // R4=0

OR R4, R2, R3 // R4=10D

LW R2, 1(R3) // R2=10D (Loaded back from memory)

ORI R4, R2, 10 // R4=10D

NOR R4, R2, R3 // R4=X“fffffff5”

SHL R4, R2, 10 // R4=X“00002800”

SHR R4, R2, 10 // R4=X“01400000”

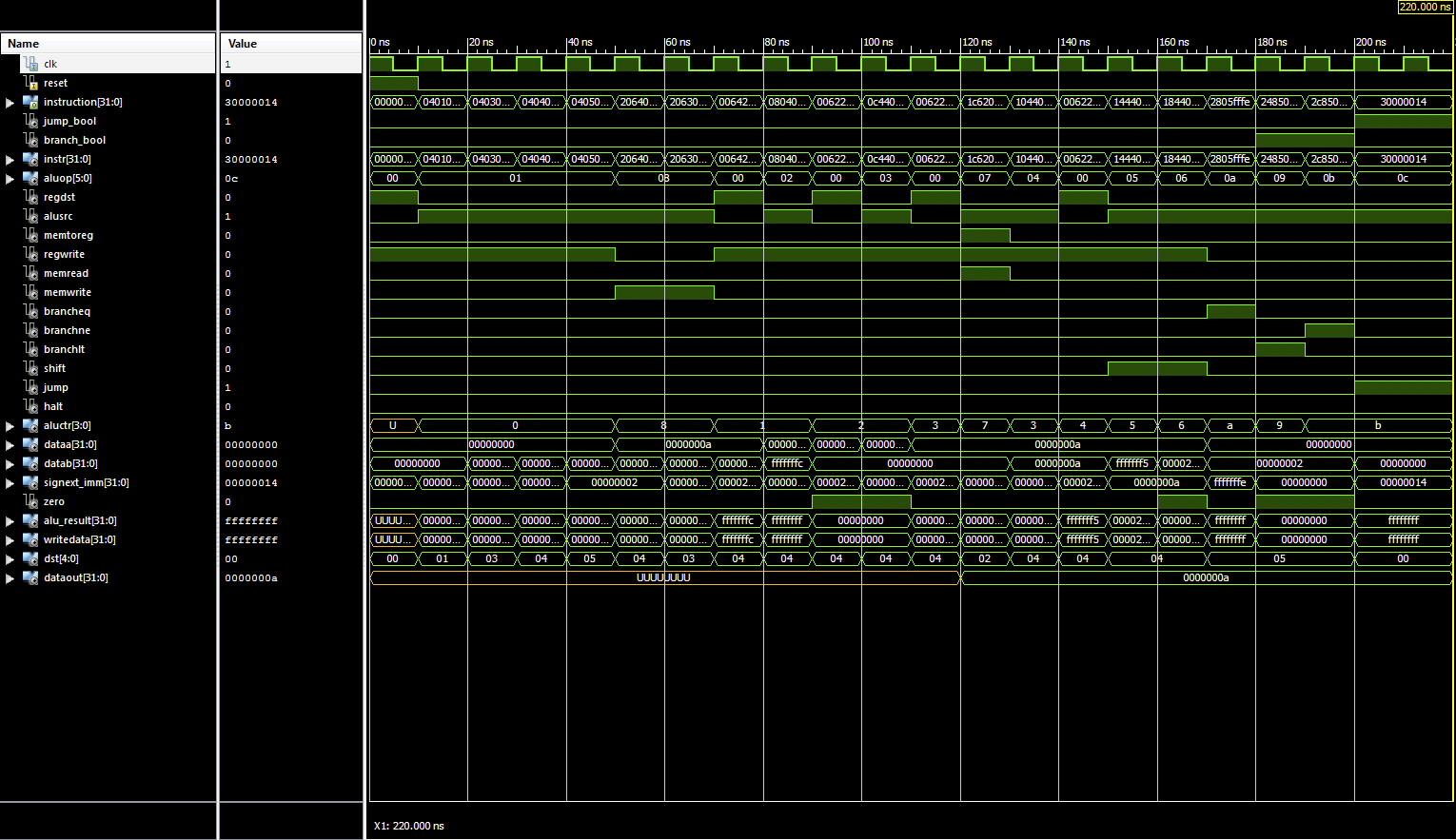
BEQ R5, R0, -2 // No Branch

BLT R5, R4, 0 // Branch to the next instruction

BNE R5, R4, 0 // Branch to the next instruction

JMP 20 // Jump

HAL // Halt



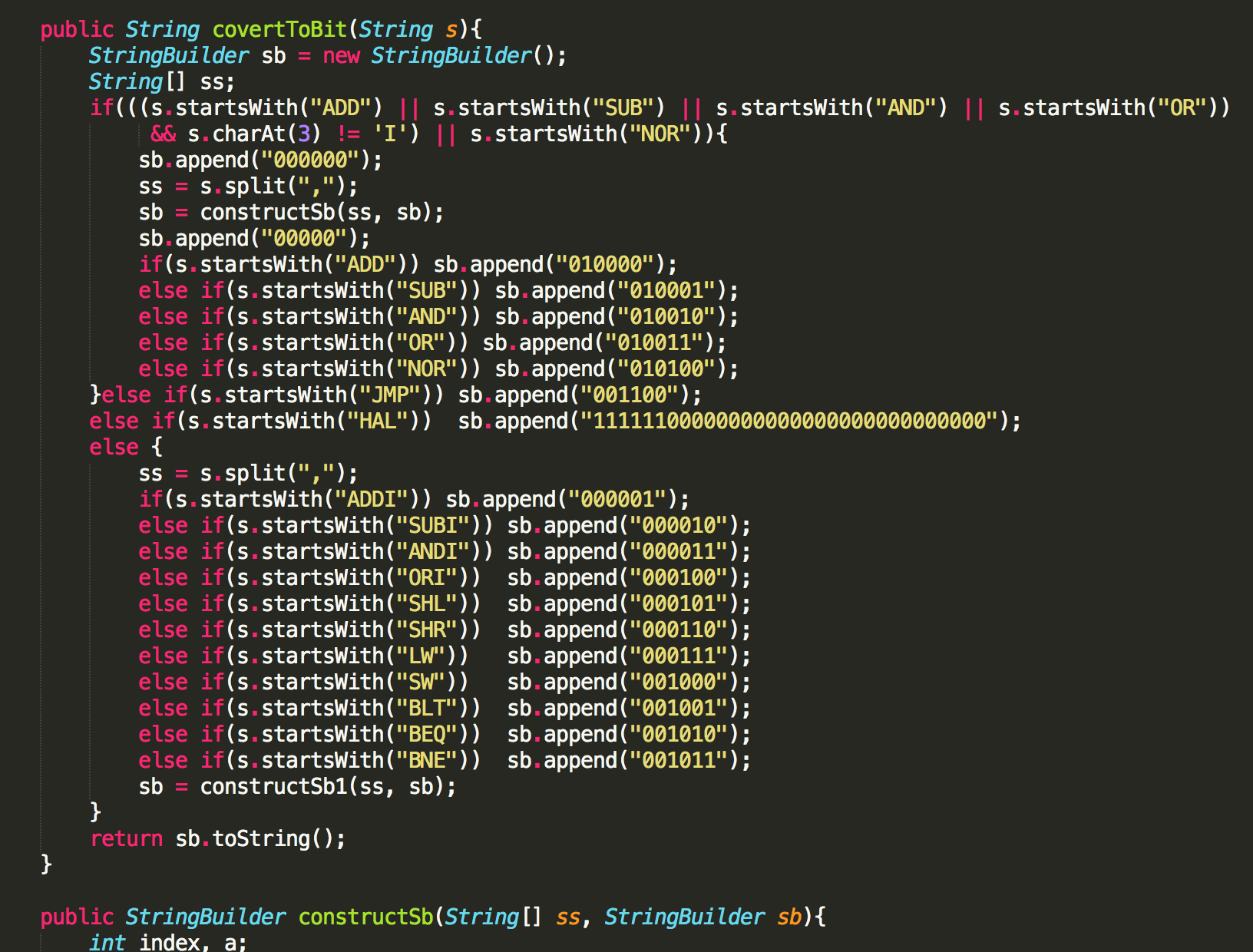
**RC5 Assembly code for Key Expansion, Encryption and Decryption**

All of our assembly code are hand written according to the given functions. Any input, output, and rotation key is stored in data memory, and any intermediate variable is temporarily stored in general register. To use inputs, we load data from memory to register via “LW” instruction. Similarly, to store outputs, we store results from register to memory via “SW” instruction. The operation parts are accord with the RC5 functions, using “ADDI”, “SUBI”, “ADD”, “OR”, “SHL”, “SHR”, etc.

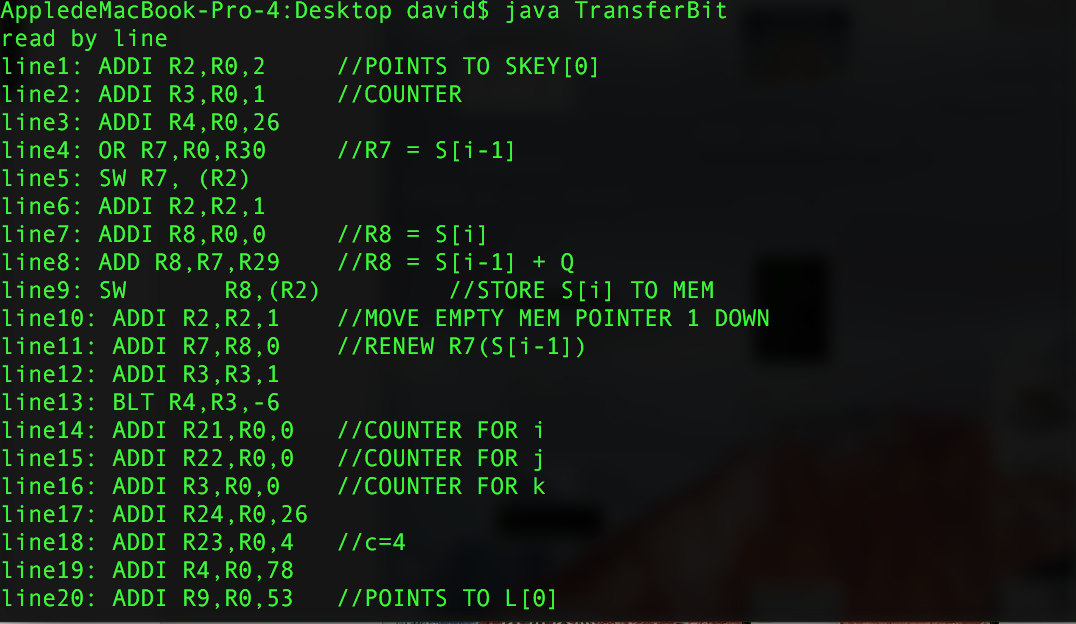
They all have loops, we use “BLT”, “BEQ”, “BNE” to take the place of loop, take “BLT” for example, when we execute “BLT Rs, Rt, -10”, if “Rs < Rt” then PC will back for 10 units, else next command will be excuted, for the rotation part, for example, when we implement “ROTL(A^B,B)” function, we used a register “Rs” to store the result of “A ^ B”, then we used “Rt” to store the result of “0x000001FF & Rb” that is the final shift number of our rotate, then we used “Rk” to store the subtract of “32 and “Rt”, then we copy “Rs” to “R1” and “R2”, then we use “SHL R1, 1” and a loop to shift “R1” left for “Rt” times and then store it in “R1”, and we use “SHR R2, 1” and a loop to shift “Rk” times and then store it in “R2”, and execute “AND Rs, R1, R2” to use “Rs” to store the result of “R1 & R2”. That is the major difficult we faced when we wrote the assembly code.

**Assembly Code to Machine Code**

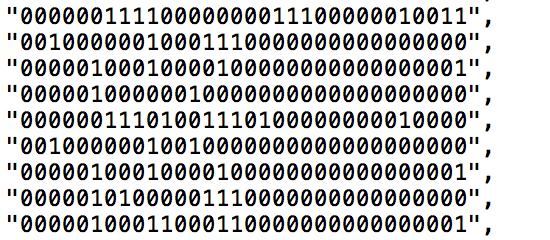
We transferred assembly code to machine code through a java program, the following is a glimpse of our program



here is a glimpse of the execution process of our program, read the assembly code line by line, and translate them line by line, then wrote the machine code to another file.



here is a glimpse of our machine code file



Complete the processor design with additional interfaces and single stepping (7-segment LEDs, Switches for input control).

High level description of how you implement and run 3 components of RC5 (encryption, decryption and key expansion) on your designed processor

Description of processor interfaces (how you control the inputs and observe the outputs)

Performance and area analysis

Details about how you verified your overall design