[CEG3420 Lab 2 Report]

|  |  |  |
| --- | --- | --- |
| [Your Name] GAO, Mingyuan |  | [SID]1155107738 |

# Task 1

**Describe the outline of how to update NEXT\_LATCHES in process\_instruction() here:**

NEXT\_LATCHES.PC = Low16bits(CURRENT\_LATCHES.PC + 2);

Next PC is the current PC+2 because there are two 8-bits word-aligned locations in each memory line pointed by the PC.

**Describe the outline of how to implement the function “memWord()” here:**

startAddr >>= 1; // startAddr /= 2; because the increment of PC is 2 each time, but increment of startAddr is only 1.

int low = MEMORY[startAddr][0];

int high = MEMORY[startAddr][1]; // save high 8 bits and low 8 bits into two variables respectively.

int content = (high << 8) + low; // combine the high 8 bits and low 8 bits to a 16-bits word

return content;

**Describe the outline of how to implement the function partVal() here:**

int tmp = Low16bits(instr << (15 - hBit)); // eliminate the bits higher than hBit

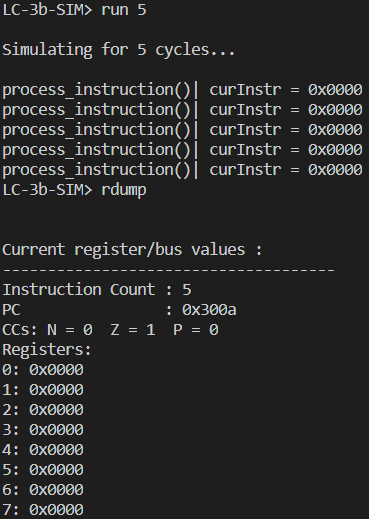
return (tmp >> (15 - hBit + lBit));

// move the variable tmp to make it start from bit0

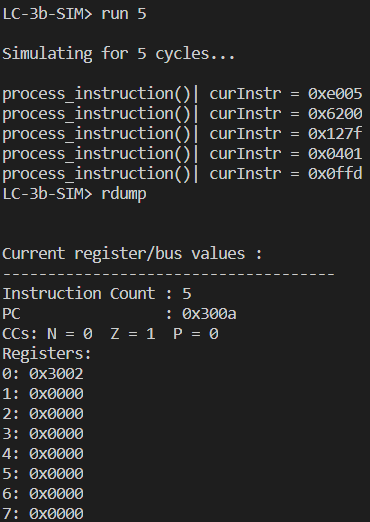
**Paste your experimental results of** **“run 5” here by running “nop.cod”, “count10.cod” and “toupper.cod” respectively.**

**(You need paste the information of instructions and registers)**

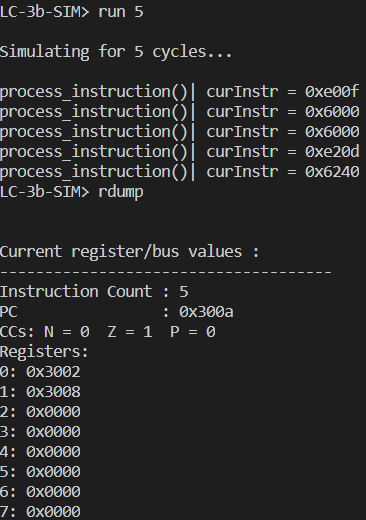
**nop.cod**



**count10.cod**



**toupper.cod**



# Task 2

**Describe the outline of how to implement function “SEXT()” here:**

imm = partVal(imm, width - 1, 0); // eliminate useless bits except for bits of imm

int leftBit = imm >> (width - 1); // get the highest bit of imm, which tells us whether imm is a negative number or a positive number

if (leftBit == 1){ // if it is a negative number, fill all bits from width-1 to 15 with 1

int tmp = 1 << width;

return Low16bits(tmp + imm); // remove useless 1s higher than bit15

}

return Low16bits(imm); // if it is non-negative, return it directly

**Describe the outline of how to implement function “setCC()” here**

if (num == 0){ // if the num is 0, set cc Z to 1; N,P to 0

NEXT\_LATCHES.Z = 1;

NEXT\_LATCHES.N = 0;

NEXT\_LATCHES.P = 0;

}

else if ((num>>15) == 0){ // if the num is not 0 and the highest bit is 0 🡪 it is a negative number, so set cc P to 1; N,Z to 0

NEXT\_LATCHES.Z = 0;

NEXT\_LATCHES.N = 0;

NEXT\_LATCHES.P = 1;

}

else if ((num>>15) == 1){ // if the highest bit of num is 1 🡪 it is a positive number, so set cc P to 1; N,Z to 0

NEXT\_LATCHES.Z = 0;

NEXT\_LATCHES.N = 1;

NEXT\_LATCHES.P = 0;

}

**Describe the influence of the different SID in this task and how you resolve it.**

My SID is 1155107738, which is an even ending. The DR in LDB is bit 6-8, BaseR in LDB is 9-11.

DR = partVal(curInstr, 8, 6);

BaseR = partVal(curInstr, 11, 9);

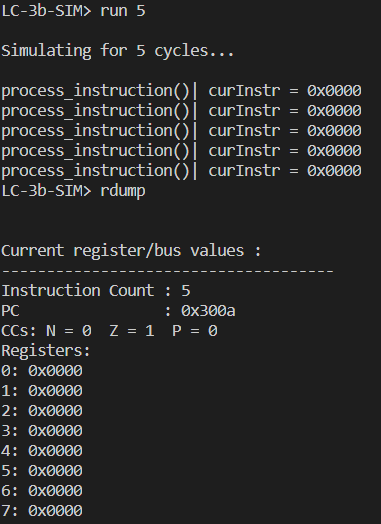
What I need to do is just to change the argumrnts hBit and lBit) passed into the partVal(), to get the correct part of curInstr.

**Paste your experimental results of “run 5” here by running**

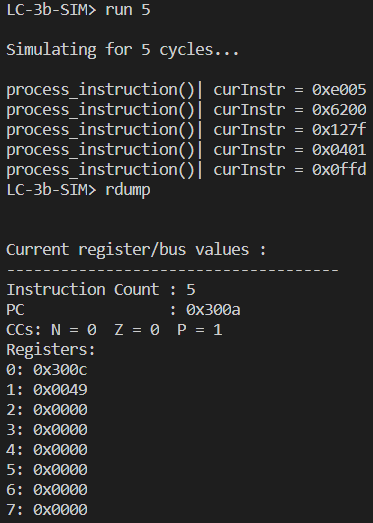
1. **for SID with even ending : “nop­­\_even.cod”, “count10\_even.cod” and “toupper2\_even.cod” respectively.**
2. **for SID with odd ending : “nop­­\_odd.cod”, “count10\_odd.cod” and “toupper2\_odd.cod” respectively.**

**(You need paste the information of instructions and registers)**

**nop­­\_even.cod**



**count10\_even.cod**



toupper2\_even.cod

