**DLXSIM ADDING INSTRUCTIONS -TUTORIAL**

1. **dlx.h:**
2. Add the new OP\_COMMAND

#define OP\_AVG 74

1. **sim.c:**
2. Add the COMMAND NAME i.e. "AVG" in operationNames[] at the location of (OP\_COMMAND-1)

char \*operationNames[] = {

/\* 0: \*/ "", "ADD", "SUB", "MUL", "DIV", "DIVU", "MOD", "MODU",

/\* 8: \*/ "AND", "NAND", "OR", "NOR", "XOR", "LLS", "LRS", "ARS",

/\* 16: \*/ "ELT", "ELTU", "EEQ", "ENEQ", "ADDI", "SUBI", "ANDI", "ORI",

/\* 24: \*/ "XORI", "LLSI", "LRSI", "ARSI", "LSOI", "LB", "LH", "LW",

/\* 32: \*/ "SB", "SH", "SW", "BRZ", "BRNZ", "JP", "JPL", "TRP", "JPR",

/\* 40: \*/ "JPRL", "NOP", "RETI", "EXBW", "EXHW", "ADDU", "ADDUI", "LHI",

/\* 48: \*/ "LBU", "LHU", "MULU", "EEQI", "EGE", "EGEI", "EGEU", "EGT",

/\* 56: \*/ "EGTI", "EGTU", "ELE", "ELEI", "ELEU", "ELTI", "ENEQI", "SUBU",

/\* 64: \*/ "SUBUI", "CMOV", "BEQ", "BNEQ", "CADD", "BGTU", "BLEU", "BLTU",

/\* 72: \*/ "BGEU", "AVG", "", "", "", "", "", "",

/\* 80: \*/ "", "", "", "", "", "", "", "",

/\* 88: \*/ "", "", "", "", "", "", "", "",

/\* 96 \*/ "", "", "", "", "", ""

};

1. **asm.c:**
2. define new command "avg", opcode/function and complete row in opcodes[] array. avg looks like add instruction. Just copy from there.

\*/

OpcodeInfo opcodes[] = {

//name class op mask other flags rangeMask

{"add", ARITH\_3PARAM, 0x001, 0x1ffff, 0x20, CHECK\_LAST|SIGN\_EXTENDED, 0xffff8000},// basic from Brownie

{"avg", ARITH\_3PARAM, 0xe81, 0x1ffff, 0x20, CHECK\_LAST|SIGN\_EXTENDED, 0xffff8000},//

1. We chose unused opcode/function as 0xe81, which results into 0x3a=function, 0x01=opcode (opcode bits: 0-5, function bits: 6-16)
2. **sim.c:**
3. Add at the location of 0x3a=function the OP\_AVG

int specialTable[] = {

/\* 0: 0x00\*/ OP\_ADD, OP\_SUB, OP\_MUL, OP\_DIV, OP\_MOD, OP\_DIVU, OP\_MODU, OP\_RES,

/\* 8: 0x08\*/ OP\_RES, OP\_RES, OP\_RES, OP\_CMOV, OP\_RES, OP\_CADD, OP\_RES, OP\_RES,

/\* 16: 0x10\*/ OP\_AND, OP\_OR, OP\_XOR, OP\_NAND, OP\_NOR, OP\_RES, OP\_RES, OP\_EGTU,

/\* 24: 0x18\*/ OP\_ELEU, OP\_EGEU, OP\_RES, OP\_RES, OP\_RES, OP\_RES, OP\_RES, OP\_RES,

/\* 32: 0x20\*/ OP\_LLS, OP\_LRS, OP\_ARS, OP\_RES, OP\_RES, OP\_RES, OP\_RES, OP\_ADDU,

/\* 40: 0x28\*/ OP\_SUBU, OP\_EGT, OP\_ELE, OP\_EGE, OP\_RES, OP\_RES, OP\_RES, OP\_RES,

/\* 48: 0x30\*/ OP\_ELT, OP\_ELTU, OP\_EEQ, OP\_ENEQ, OP\_RES, OP\_RES, OP\_RES, OP\_RES,

/\* 56: 0x38\*/ OP\_RES, OP\_RES, OP\_AVG, OP\_RES, OP\_RES, OP\_RES, OP\_RES, OP\_RES

};

1. Add the new case for implementation of avg under OP\_AVG:

case OP\_AVG:

LoadRegisterS1 LoadRegisterS2

writeRegister(machPtr, wordPtr->rd, (rs1+rs2)/2, 0);

break;

1. Test using a custom assembly program (see below).

-bash-3.2$ ./dlxsim -f/home/sajjad/SS20/ASIPMeisterProjects/1/brownie/Applications/testAVG.dlxsim -da0 -pf1

Biggest used address for Text Section (word aligned): 0x4c

Biggest used address for Data Section (word aligned): 0x0

(dlxsim) go

TRAP #0 received

Altogether 34,0e0(34) cycles executed.

0 Warnings for unresolved data dependencies printed.

0 Warnings for successive load/store commands printed.

0 Warnings for load/stores in the text section printed.

(dlxsim) get r21

r21: 0x0000000f

(dlxsim) get r22

r22: 0x00000009

(dlxsim) get r24

r24: 0x00000009

(dlxsim) get r25

r25: 0x00000006

(dlxsim) get r26

r26: 0x0000000c

(dlxsim) get r21 d

r21: 15

(dlxsim) get r22 d

r22: 9

(dlxsim) get r24 d

r24: 9

(dlxsim) get r25 d

r25: 6

(dlxsim) get r26 d

r26: 12

(dlxsim) q

-bash-3.2$

1. **testAVG.s**
2. Preparing the test program, take 1\_Arith.s and save it as testAVG.s and insert AVG instruction:

.text

.align 2

.globl \_main

.type \_main, @function

\_main:

ADDI R21, R0, #5

ADDI R22, R0, #9

AND R24, R21, R22

SUB R25, R21, R22

AVG R26, R21, R22

jpr r3 ; return

.size \_main, .-\_main

.ident "GCC: (GNU) 4.2.2"

1. Delete everything before \_main and append the following at the top before the \_main:

.section .text

.addressing Word

xor r4, r4, r4

xor r5, r5, r5

xor r6, r6, r6

addi r4, r0, $(0x0017EFFC / 0x10000)

addi r5, r0, $(0x0017EFFC / 0x10000)

lsoi r4, r4, $(0x0017EFFC % 0x10000)

lsoi r5, r5, $(0x0017EFFC % 0x10000)

sw -4(r5), r4

sw -8(r5), r3

addi r5, r4, $(-8)

jpl \_main

sw 0(r5), r21

jpl \_exit

.section .text

.addressing .Word

; .align 2

; .globl \_main

; .type \_main, @function

1. Delete everything after “jpr r3; return” and insert the following at the bottom:

; .size \_main, .-\_main

; .ident "GCC: (GNU) 4.2.2"

\_exit:

trap #0 ; HALT

1. testAVG.dlxsim will looks like:

.section .text

.addressing Word

xor r4, r4, r4

xor r5, r5, r5

xor r6, r6, r6

addi r4, r0, $(0x0017EFFC / 0x10000)

addi r5, r0, $(0x0017EFFC / 0x10000)

lsoi r4, r4, $(0x0017EFFC % 0x10000)

lsoi r5, r5, $(0x0017EFFC % 0x10000)

sw -4(r5), r4

sw -8(r5), r3

addi r5, r4, $(-8)

jpl \_main

sw 0(r5), r21

jpl \_exit

.section .text

.addressing .Word

; .align 2

; .globl \_main

; .type \_main, @function

\_main:

ADDI R21, R0, #15

ADDI R22, R0, #9

AND R24, R21, R22

SUB R25, R21, R22

AVG R26, R21, R22

jpr r3 ; return

; .size \_main, .-\_main

; .ident "GCC: (GNU) 4.2.2"

\_exit:

trap #0 ; HALT