

# Tutorial - DLXSIM Adding Instructions

## Customized Embedded Processor Design

Application Specific Instruction-Set Processors- ASIP  
Lab (Praktikum)

Responsible/Author:  
**MSc. Sajjad Hussain**

**Supervisors:**

MSc. Sajjad Hussain, Dr.-Ing. Lars Bauer, Prof. Dr.-Ing. Jörg Henkel

Chair of Embedded Systems,  
Building 07.21, Haid-und-Neu-Str. 7,  
76131 Karlsruhe, Germany.

August 7, 2022

# DLXSIM ADDING INSTRUCTIONS -TUTORIAL

## Step to Add a Custom Instruction

Follow the steps below and edit the mentioned files one by one.

### A. dlx.h::

1. Add the new OP\_COMMAND

```
\#define OP\_AVG 74
```

### B. 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: */ "", "", "", "", "", "", "", ""
};
```

### C. asm.c::

3. 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", ARITH3PARAM, 0x001, 0x1ffff, 0x20, CHECK_LAST|
    //      SIGN_EXTENDED, 0xffff8000
    //      }, // basic from Brownie
    {"avg", ARITH3PARAM, 0xe81, 0x1ffff, 0x20, CHECK_LAST|
    //      SIGN_EXTENDED, 0xffff8000
    //      }, //

```

4. We chose unused opcode/function as 0xe81, which results into 0x3a=function, 0x01=opcode (opcode bits: 0-5, function bits: 6-16)

**D. sim.c::**

5. Add at the location of 0x3a=function the OP\_AVG

```

/* 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
};

```

6. 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;

```

7. compile the dlxsim using "make".

8. 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$

```

**E. testAVG.s:**

9. 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"

```

10. 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, $(0x0017EFFF / 0x10000)
addi     r5, r0, $(0x0017EFFF / 0x10000)

lsoi     r4, r4, $(0x0017EFFF % 0x10000)
lsoi     r5, r5, $(0x0017EFFF % 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

```

11. 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

```

12. `testAVG.dlx` will look like:

```

        .section          .text
        .addressing       Word
xor      r4, r4, r4
xor      r5, r5, r5
xor      r6, r6, r6
addi     r4, r0, $(0x0017EFFF / 0x10000)
addi     r5, r0, $(0x0017EFFF / 0x10000)

lsoi     r4, r4, $(0x0017EFFF % 0x10000)
lsoi     r5, r5, $(0x0017EFFF % 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
```