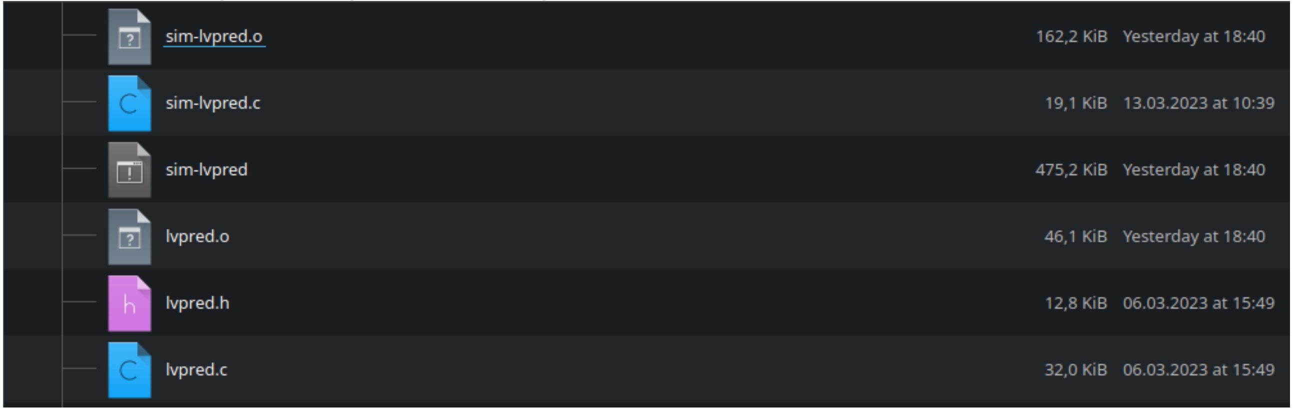
Tema 2

Bucur Dan-Alexandru (243/2)
&
Timar Cosmin (243/1)

Conform cerintelor temei 2 cat si a laboratorului 1.3 (https://webspace.ulbsibiu.ro/adrian.florea/html/simulatoare/L03_Extinderea_Simplescalar.pdf)

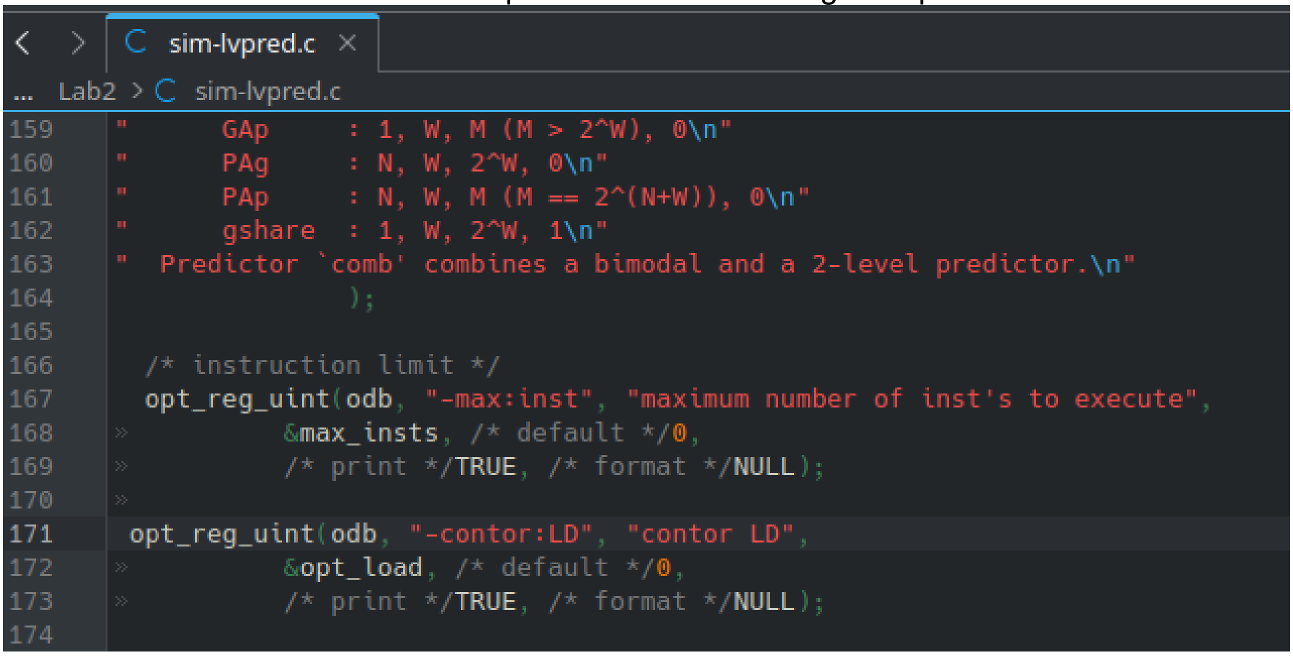
Am avut de implemtat parametrizarile date:
1. (-contor:LD 0/1) cu acesta vom afla cate instructiuni cu referire la memorie avem din categoria load cat si store. (done)

Folosindu-ne de fiserele bpred.c, bpred.h si sim-bpred.c ale simulatorului sim-bpred am putut crea simulatorul lv-pred, copiind aceste componente si redenumindu-le de asemenea la lvpred.c, lvpred.h, sim-lvpred.c.

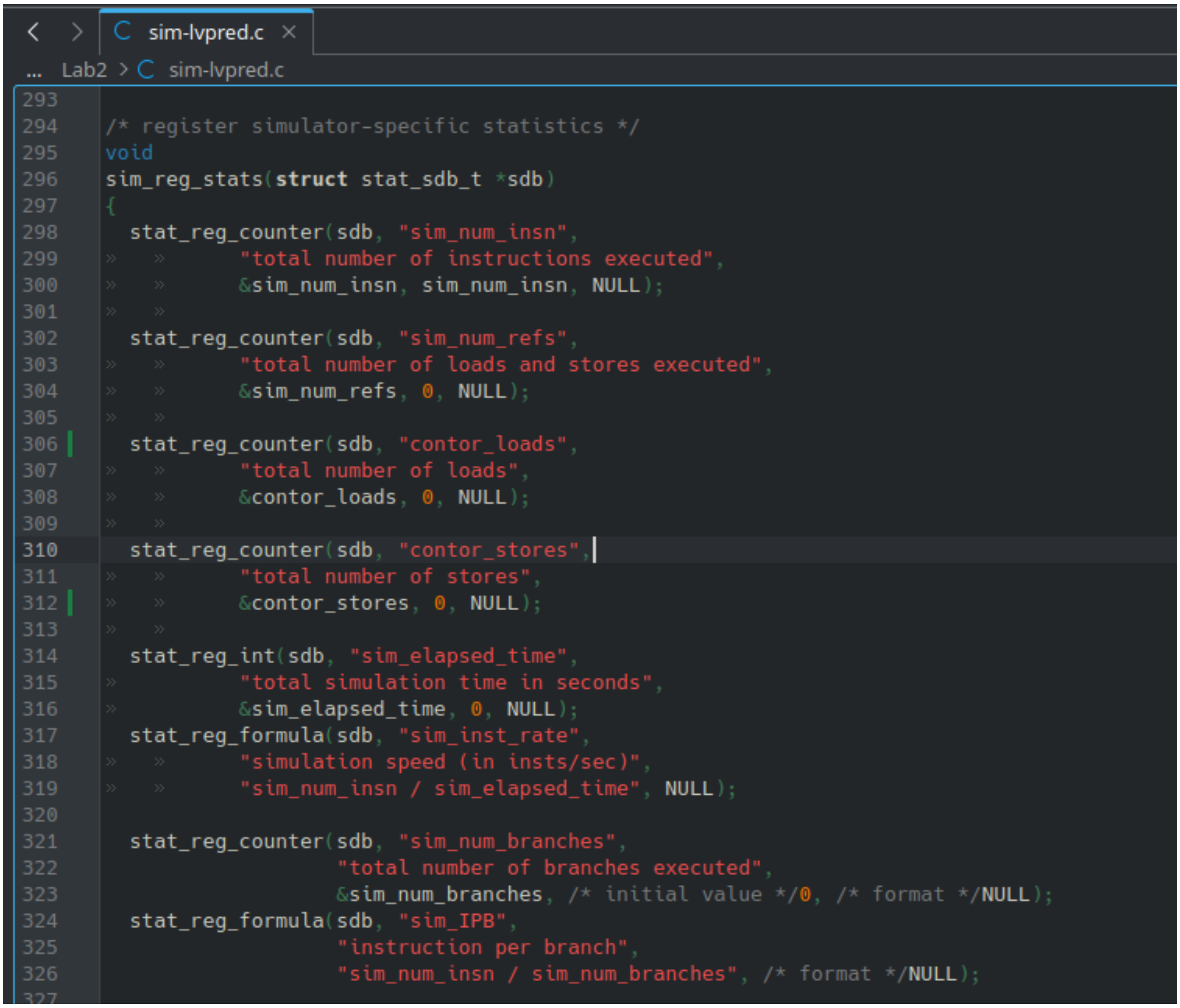


sim-lvpred.o	162,2 KiB	Yesterday at 18:40
sim-lvpred.c	19,1 KiB	13.03.2023 at 10:39
sim-lvpred	475,2 KiB	Yesterday at 18:40
lvpred.o	46,1 KiB	Yesterday at 18:40
lvpred.h	12,8 KiB	06.03.2023 at 15:49
lvpred.c	32,0 KiB	06.03.2023 at 15:49

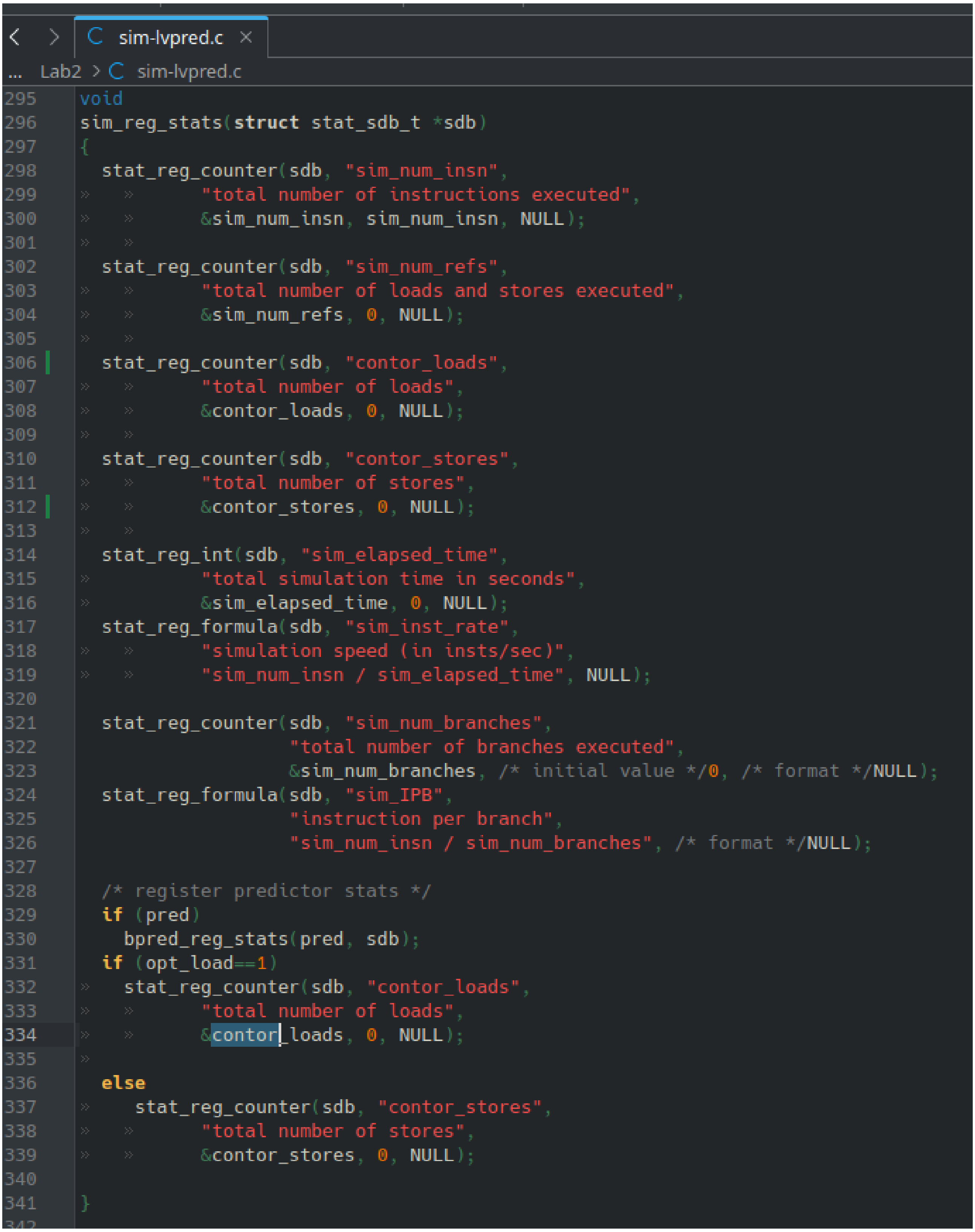
De asemenea am editat fisierele copiate si redumite adaugand optiunea de contor:



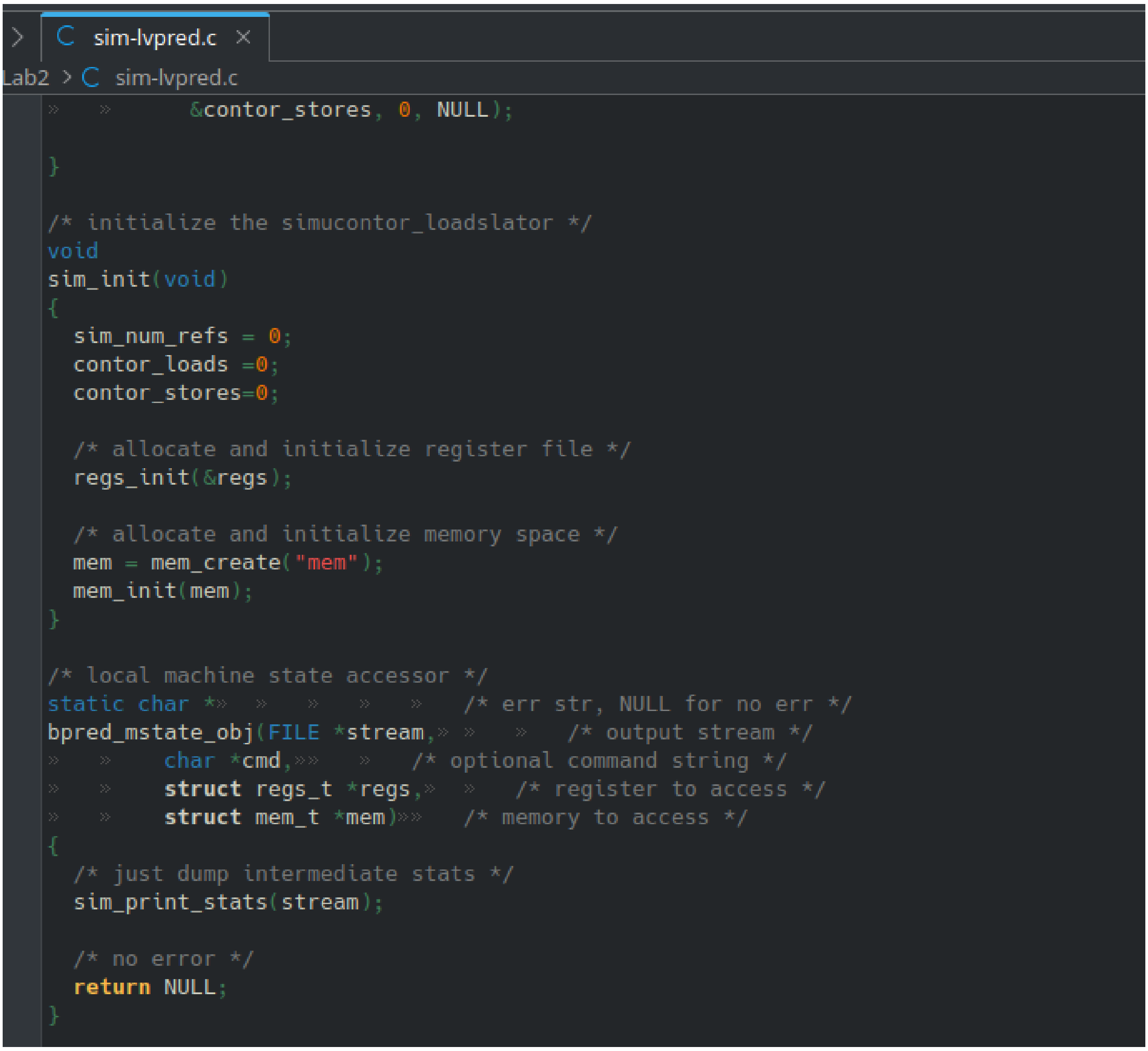
```
"      GAp     : 1, W, M (M > 2^W), 0\n"
"      PAg     : N, W, 2^W, 0\n"
"      PAp     : N, W, M (M == 2^(N+W)), 0\n"
"      gshare  : 1, W, 2^W, 1\n"
"  Predictor `comb' combines a bimodal and a 2-level predictor.\n"
               );

  /* instruction limit */
  opt_reg_uint(odb, "-max:inst", "maximum number of inst's to execute",
         &max_insts, /* default */0,
         /* print */TRUE, /* format */NULL);

 opt_reg_uint(odb, "-contor:LD", "contor LD",
         &opt_load, /* default */0,
         /* print */TRUE, /* format */NULL);
```



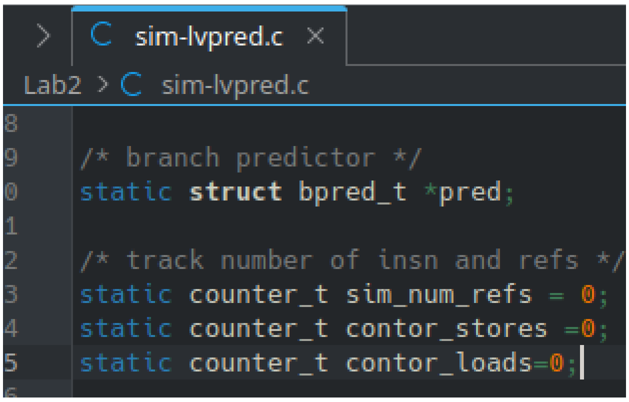
```c

/* register simulator-specific statistics */
void
sim_reg_stats(struct stat_sdb_t *sdb)
{
  stat_reg_counter(sdb, "sim_num_insn",
           "total number of instructions executed",
           &sim_num_insn, sim_num_insn, NULL);

  stat_reg_counter(sdb, "sim_num_refs",
           "total number of loads and stores executed",
           &sim_num_refs, 0, NULL);

  stat_reg_counter(sdb, "contor_loads",
           "total number of loads",
           &contor_loads, 0, NULL);

  stat_reg_counter(sdb, "contor_stores",
           "total number of stores",
           &contor_stores, 0, NULL);

  stat_reg_int(sdb, "sim_elapsed_time",
           "total simulation time in seconds",
           &sim_elapsed_time, 0, NULL);
  stat_reg_formula(sdb, "sim_inst_rate",
           "simulation speed (in insts/sec)",
           "sim_num_insn / sim_elapsed_time", NULL);

  stat_reg_counter(sdb, "sim_num_branches",
                   "total number of branches executed",
                   &sim_num_branches, /* initial value */0, /* format */NULL);
  stat_reg_formula(sdb, "sim_IPB",
                   "instruction per branch",
                   "sim_num_insn / sim_num_branches", /* format */NULL);
```



```c
void
sim_reg_stats(struct stat_sdb_t *sdb)
{
  stat_reg_counter(sdb, "sim_num_insn",
           "total number of instructions executed",
           &sim_num_insn, sim_num_insn, NULL);

  stat_reg_counter(sdb, "sim_num_refs",
           "total number of loads and stores executed",
           &sim_num_refs, 0, NULL);

  stat_reg_counter(sdb, "contor_loads",
           "total number of loads",
           &contor_loads, 0, NULL);

  stat_reg_counter(sdb, "contor_stores",
           "total number of stores",
           &contor_stores, 0, NULL);

  stat_reg_int(sdb, "sim_elapsed_time",
           "total simulation time in seconds",
           &sim_elapsed_time, 0, NULL);
  stat_reg_formula(sdb, "sim_inst_rate",
           "simulation speed (in insts/sec)",
           "sim_num_insn / sim_elapsed_time", NULL);

  stat_reg_counter(sdb, "sim_num_branches",
                   "total number of branches executed",
                   &sim_num_branches, /* initial value */0, /* format */NULL);
  stat_reg_formula(sdb, "sim_IPB",
                   "instruction per branch",
                   "sim_num_insn / sim_num_branches", /* format */NULL);

  /* register predictor stats */
  if (pred)
    bpred_reg_stats(pred, sdb);
  if (opt_load==1)
    stat_reg_counter(sdb, "contor_loads",
           "total number of loads",
           &contor_loads, 0, NULL);

  else
     stat_reg_counter(sdb, "contor_stores",
           "total number of stores",
           &contor_stores, 0, NULL);

}
```

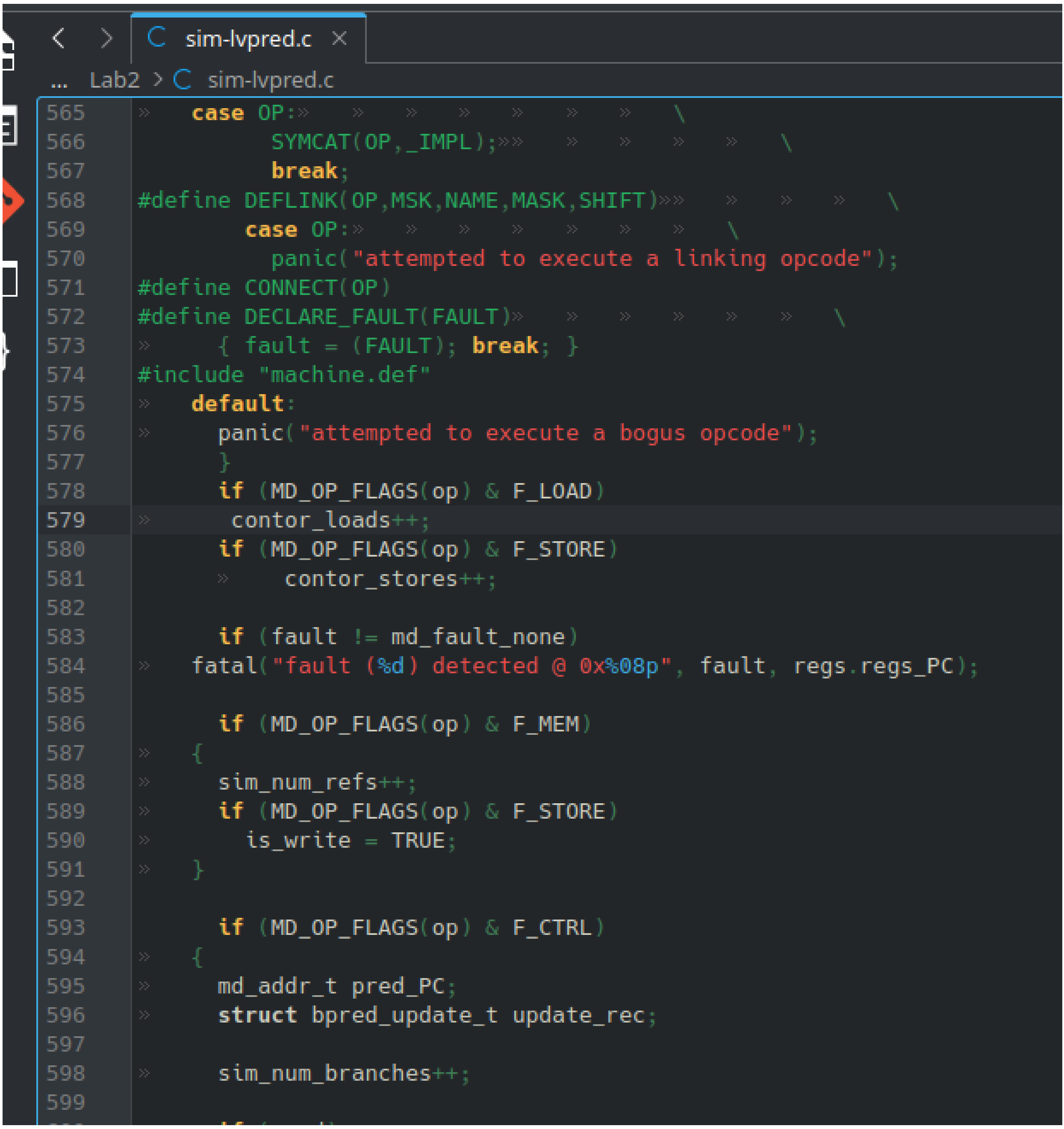


```
                &contor_stores, 0, NULL);

}

/* initialize the simucontor_loadslator */
void
sim_init(void)
{
  sim_num_refs = 0;
  contor_loads =0;
  contor_stores=0;

  /* allocate and initialize register file */
  regs_init(&regs);

  /* allocate and initialize memory space */
  mem = mem_create("mem");
  mem_init(mem);
}

/* local machine state accessor */
static char *                           /* err str, NULL for no err */
bpred_mstate_obj(FILE *stream,          /* output stream */
                 char *cmd,             /* optional command string */
                 struct regs_t *regs,   /* register to access */
                 struct mem_t *mem)     /* memory to access */
{
  /* just dump intermediate stats */
  sim_print_stats(stream);

  /* no error */
  return NULL;
}
```



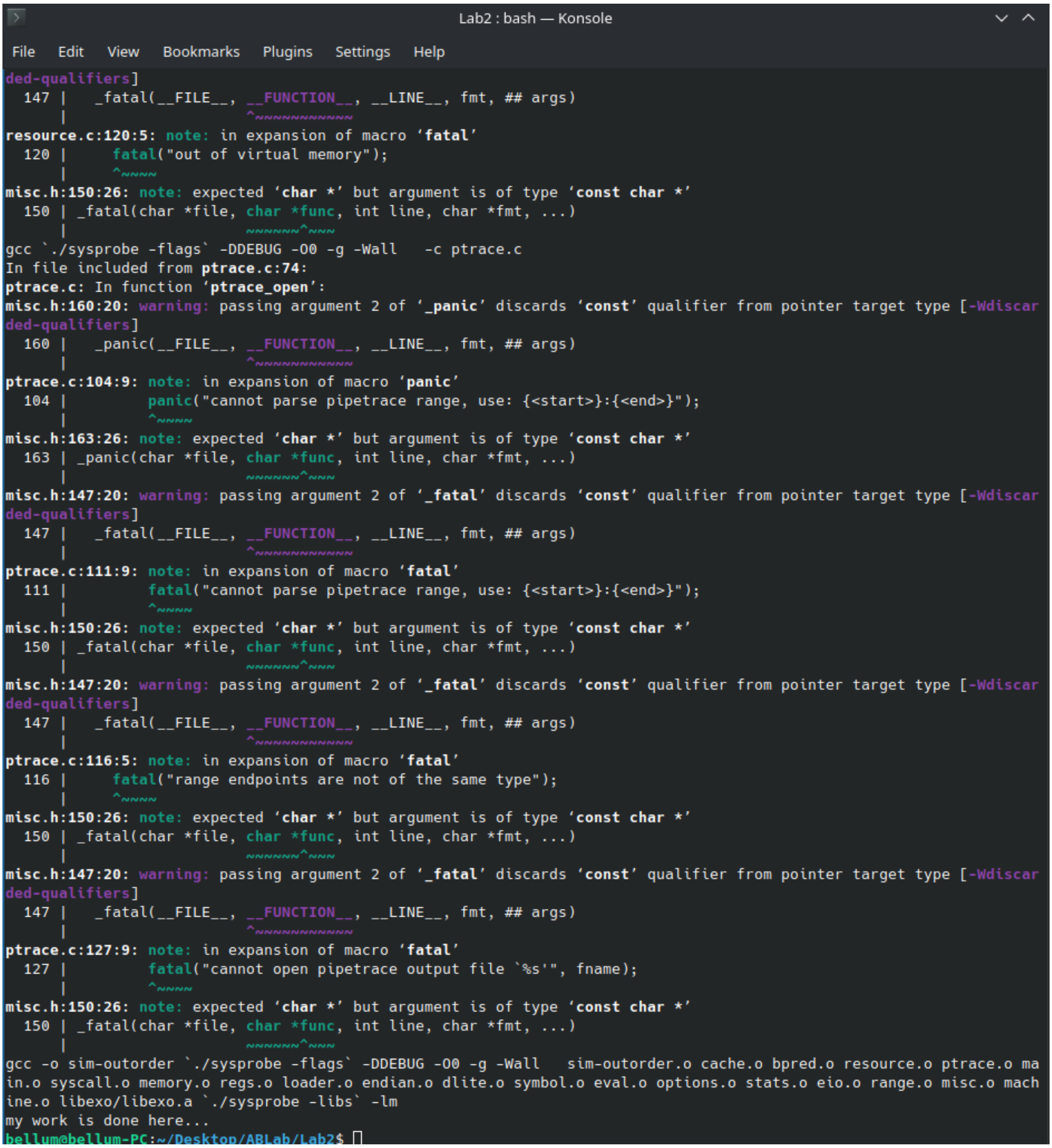
```

/* branch predictor */
static struct bpred_t *pred;

/* track number of insn and refs */
static counter_t sim_num_refs = 0;
static counter_t contor_stores =0;
static counter_t contor_loads=0;
```

lar
in

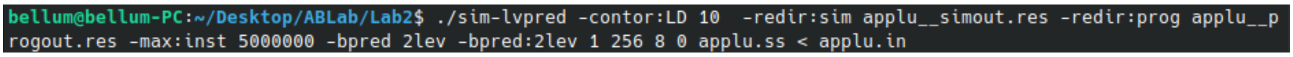


```
    case OP:                                        \
          SYMCAT(OP,_IMPL);                         \
          break;
#define DEFLINK(OP,MSK,NAME,MASK,SHIFT)             \
        case OP:                                    \
          panic("attempted to execute a linking opcode");
#define CONNECT(OP)
#define DECLARE_FAULT(FAULT)                        \
      { fault = (FAULT); break; }
#include "machine.def"
    default:
      panic("attempted to execute a bogus opcode");
      }
      if (MD_OP_FLAGS(op) & F_LOAD)
       contor_loads++;
      if (MD_OP_FLAGS(op) & F_STORE)
           contor_stores++;

      if (fault != md_fault_none)
    fatal("fault (%d) detected @ 0x%08p", fault, regs.regs_PC);

      if (MD_OP_FLAGS(op) & F_MEM)
    {
      sim_num_refs++;
      if (MD_OP_FLAGS(op) & F_STORE)
        is_write = TRUE;
    }

      if (MD_OP_FLAGS(op) & F_CTRL)
    {
      md_addr_t pred_PC;
      struct bpred_update_t update_rec;

      sim_num_branches++;

```

urma compilarii putem rula simulatorul lv-pred:



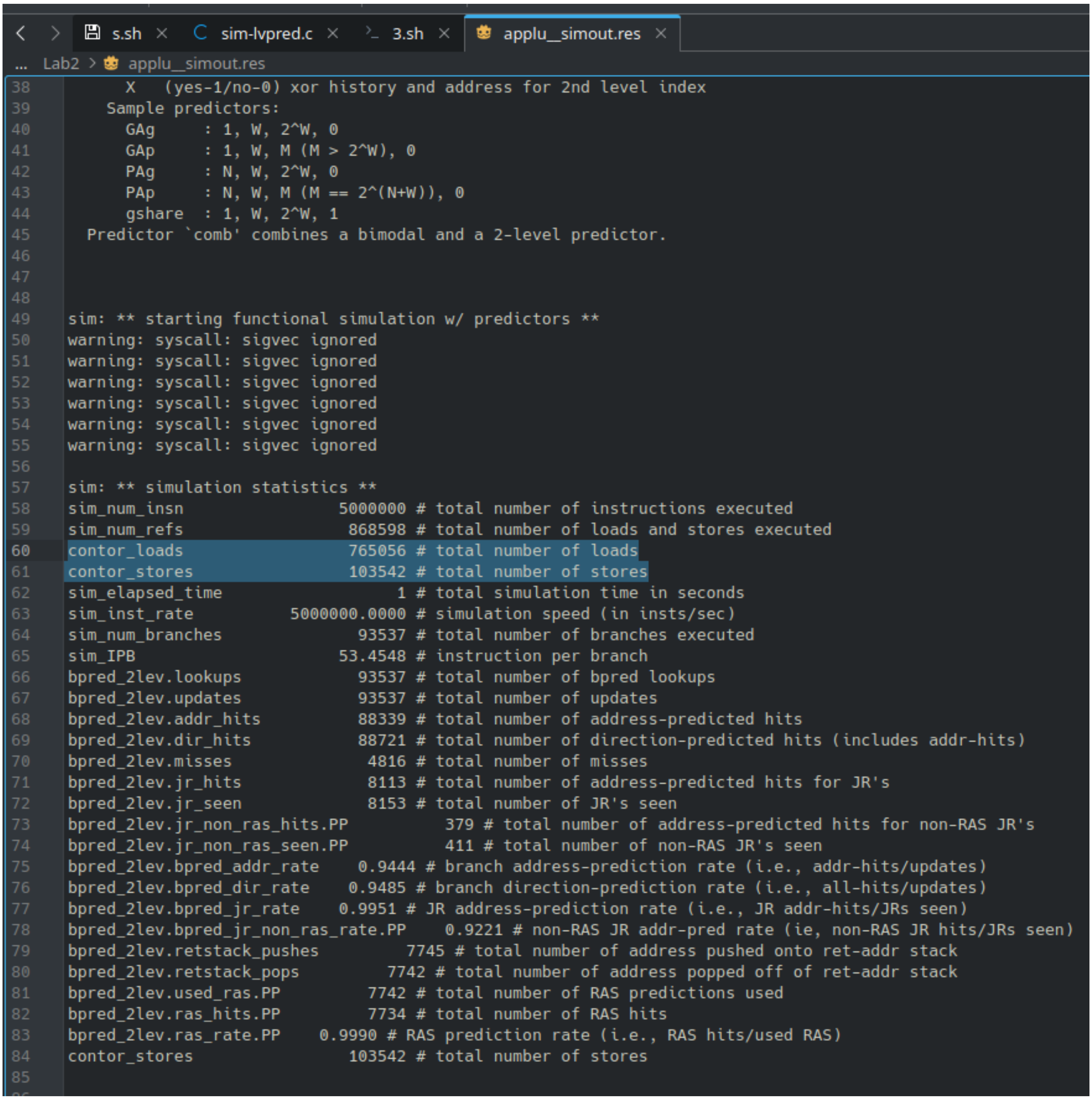
```
ded-qualifiers]
  147 |   _fatal(__FILE__, __FUNCTION__, __LINE__, fmt, ## args)
      |                    ^~~~~~~~~~~~
resource.c:120:5: note: in expansion of macro ‘fatal’
  120 |     fatal("out of virtual memory");
      |     ^~~~~
misc.h:150:26: note: expected ‘char *’ but argument is of type ‘const char *’
  150 | _fatal(char *file, char *func, int line, char *fmt, ...)
      |                    ~~~~~~^~~~
gcc `./sysprobe -flags` -DDEBUG -O0 -g -Wall   -c ptrace.c
In file included from ptrace.c:74:
ptrace.c: In function ‘ptrace_open’:
misc.h:160:20: warning: passing argument 2 of ‘_panic’ discards ‘const’ qualifier from pointer target type [-Wdiscar
ded-qualifiers]
  160 |   _panic(__FILE__, __FUNCTION__, __LINE__, fmt, ## args)
      |                    ^~~~~~~~~~~~
ptrace.c:104:9: note: in expansion of macro ‘panic’
  104 |         panic("cannot parse pipetrace range, use: {<start>}:{<end>}");
      |         ^~~~~
misc.h:163:26: note: expected ‘char *’ but argument is of type ‘const char *’
  163 | _panic(char *file, char *func, int line, char *fmt, ...)
      |                    ~~~~~~^~~~
misc.h:147:20: warning: passing argument 2 of ‘_fatal’ discards ‘const’ qualifier from pointer target type [-Wdiscar
ded-qualifiers]
  147 |   _fatal(__FILE__, __FUNCTION__, __LINE__, fmt, ## args)
      |                    ^~~~~~~~~~~~
ptrace.c:111:9: note: in expansion of macro ‘fatal’
  111 |         fatal("cannot parse pipetrace range, use: {<start>}:{<end>}");
      |         ^~~~~
misc.h:150:26: note: expected ‘char *’ but argument is of type ‘const char *’
  150 | _fatal(char *file, char *func, int line, char *fmt, ...)
      |                    ~~~~~~^~~~
misc.h:147:20: warning: passing argument 2 of ‘_fatal’ discards ‘const’ qualifier from pointer target type [-Wdiscar
ded-qualifiers]
  147 |   _fatal(__FILE__, __FUNCTION__, __LINE__, fmt, ## args)
      |                    ^~~~~~~~~~~~
ptrace.c:116:5: note: in expansion of macro ‘fatal’
  116 |     fatal("range endpoints are not of the same type");
      |     ^~~~~
misc.h:150:26: note: expected ‘char *’ but argument is of type ‘const char *’
  150 | _fatal(char *file, char *func, int line, char *fmt, ...)
      |                    ~~~~~~^~~~
misc.h:147:20: warning: passing argument 2 of ‘_fatal’ discards ‘const’ qualifier from pointer target type [-Wdiscar
ded-qualifiers]
  147 |   _fatal(__FILE__, __FUNCTION__, __LINE__, fmt, ## args)
      |                    ^~~~~~~~~~~~
ptrace.c:127:9: note: in expansion of macro ‘fatal’
  127 |         fatal("cannot open pipetrace output file `%s'", fname);
      |         ^~~~~
misc.h:150:26: note: expected ‘char *’ but argument is of type ‘const char *’
  150 | _fatal(char *file, char *func, int line, char *fmt, ...)
      |                    ~~~~~~^~~~
gcc -o sim-outorder `./sysprobe -flags` -DDEBUG -O0 -g -Wall   sim-outorder.o cache.o bpred.o resource.o ptrace.o ma
in.o syscall.o memory.o regs.o loader.o endian.o dlite.o symbol.o eval.o options.o stats.o eio.o range.o misc.o mach
ine.o libexo/libexo.a `./sysprobe -libs` -lm
my work is done here...
bellum@bellum-PC:~/Desktop/ABLab/Lab2$ 
```



```
bellum@bellum-PC:~/Desktop/ABLab/Lab2$ ./sim-lvpred -contor:LD 10  -redir:sim applu__simout.res -redir:prog applu__p
rogout.res -max:inst 5000000 -bpred 2lev -bpred:2lev 1 256 8 0 applu.ss < applu.in
```

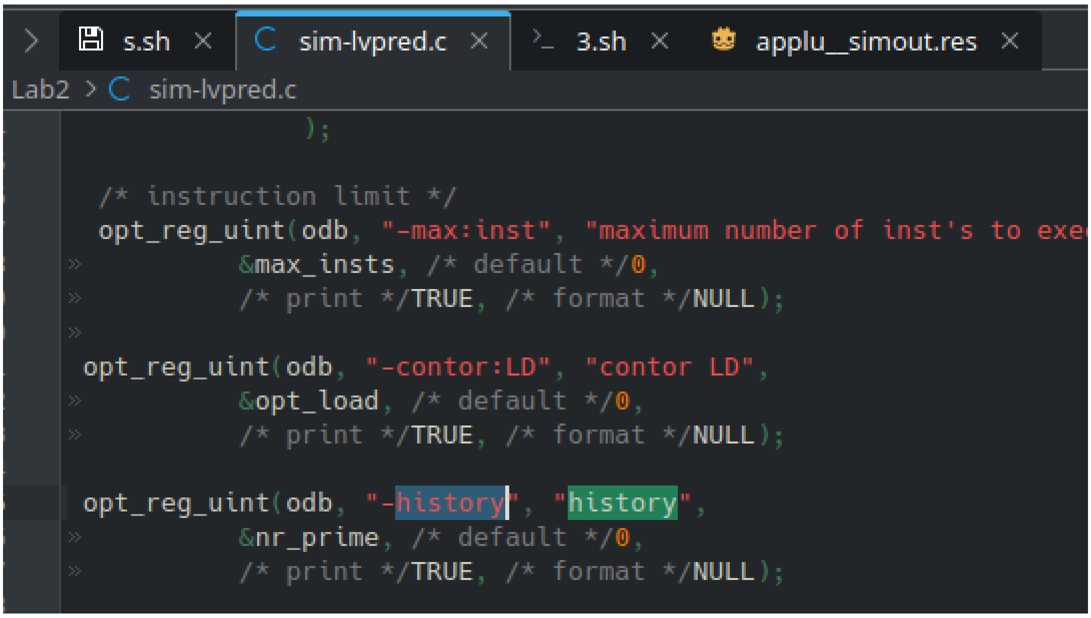
Comanda: ./sim-lvpred -contor:LD 10 -redir:sim applu__simout.res -redir:prog applu__progout.res -max:inst 5000000 -bpred 2lev -bpred:2lev 1 256 8 0 applu.ss < applu.in

Iar rezultatul este:

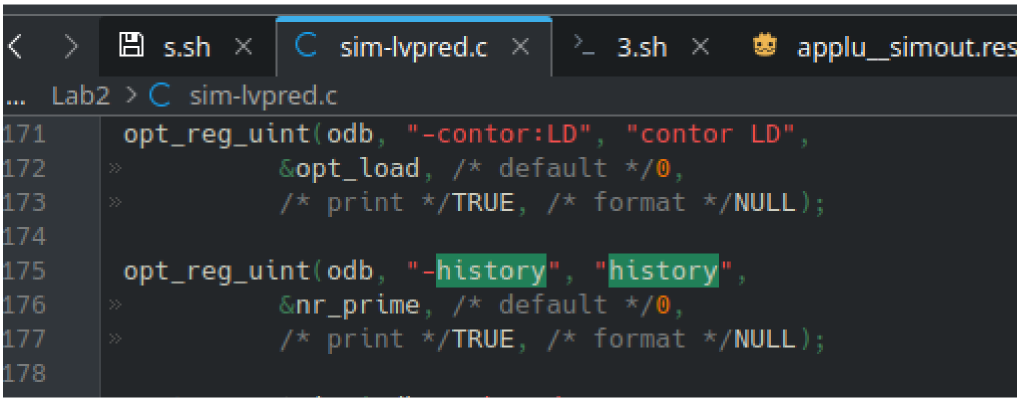


```
      X   (yes-1/no-0) xor history and address for 2nd level index
    Sample predictors:
      GAg     : 1, W, 2^W, 0
      GAp     : 1, W, M (M > 2^W), 0
      PAg     : N, W, 2^W, 0
      PAp     : N, W, M (M == 2^(N+W)), 0
      gshare  : 1, W, 2^W, 1
  Predictor `comb' combines a bimodal and a 2-level predictor.



sim: ** starting functional simulation w/ predictors **
warning: syscall: sigvec ignored
warning: syscall: sigvec ignored
warning: syscall: sigvec ignored
warning: syscall: sigvec ignored
warning: syscall: sigvec ignored
warning: syscall: sigvec ignored

sim: ** simulation statistics **
sim_num_insn                5000000 # total number of instructions executed
sim_num_refs                 868598 # total number of loads and stores executed
contor_loads                 765056 # total number of loads
contor_stores                103542 # total number of stores
sim_elapsed_time                  1 # total simulation time in seconds
sim_inst_rate          5000000.0000 # simulation speed (in insts/sec)
sim_num_branches              93537 # total number of branches executed
sim_IPB                     53.4548 # instruction per branch
bpred_2lev.lookups            93537 # total number of bpred lookups
bpred_2lev.updates            93537 # total number of updates
bpred_2lev.addr_hits          88339 # total number of address-predicted hits
bpred_2lev.dir_hits           88721 # total number of direction-predicted hits (includes addr-hits)
bpred_2lev.misses              4816 # total number of misses
bpred_2lev.jr_hits             8113 # total number of address-predicted hits for JR's
bpred_2lev.jr_seen             8153 # total number of JR's seen
bpred_2lev.jr_non_ras_hits.PP         379 # total number of address-predicted hits for non-RAS JR's
bpred_2lev.jr_non_ras_seen.PP         411 # total number of non-RAS JR's seen
bpred_2lev.bpred_addr_rate    0.9444 # branch address-prediction rate (i.e., addr-hits/updates)
bpred_2lev.bpred_dir_rate    0.9485 # branch direction-prediction rate (i.e., all-hits/updates)
bpred_2lev.bpred_jr_rate    0.9951 # JR address-prediction rate (i.e., JR addr-hits/JRs seen)
bpred_2lev.bpred_jr_non_ras_rate.PP    0.9221 # non-RAS JR addr-pred rate (ie, non-RAS JR hits/JRs seen)
bpred_2lev.retstack_pushes         7745 # total number of address pushed onto ret-addr stack
bpred_2lev.retstack_pops         7742 # total number of address popped off of ret-addr stack
bpred_2lev.used_ras.PP         7742 # total number of RAS predictions used
bpred_2lev.ras_hits.PP         7734 # total number of RAS hits
bpred_2lev.ras_rate.PP    0.9990 # RAS prediction rate (i.e., RAS hits/used RAS)
contor_stores                103542 # total number of stores

```

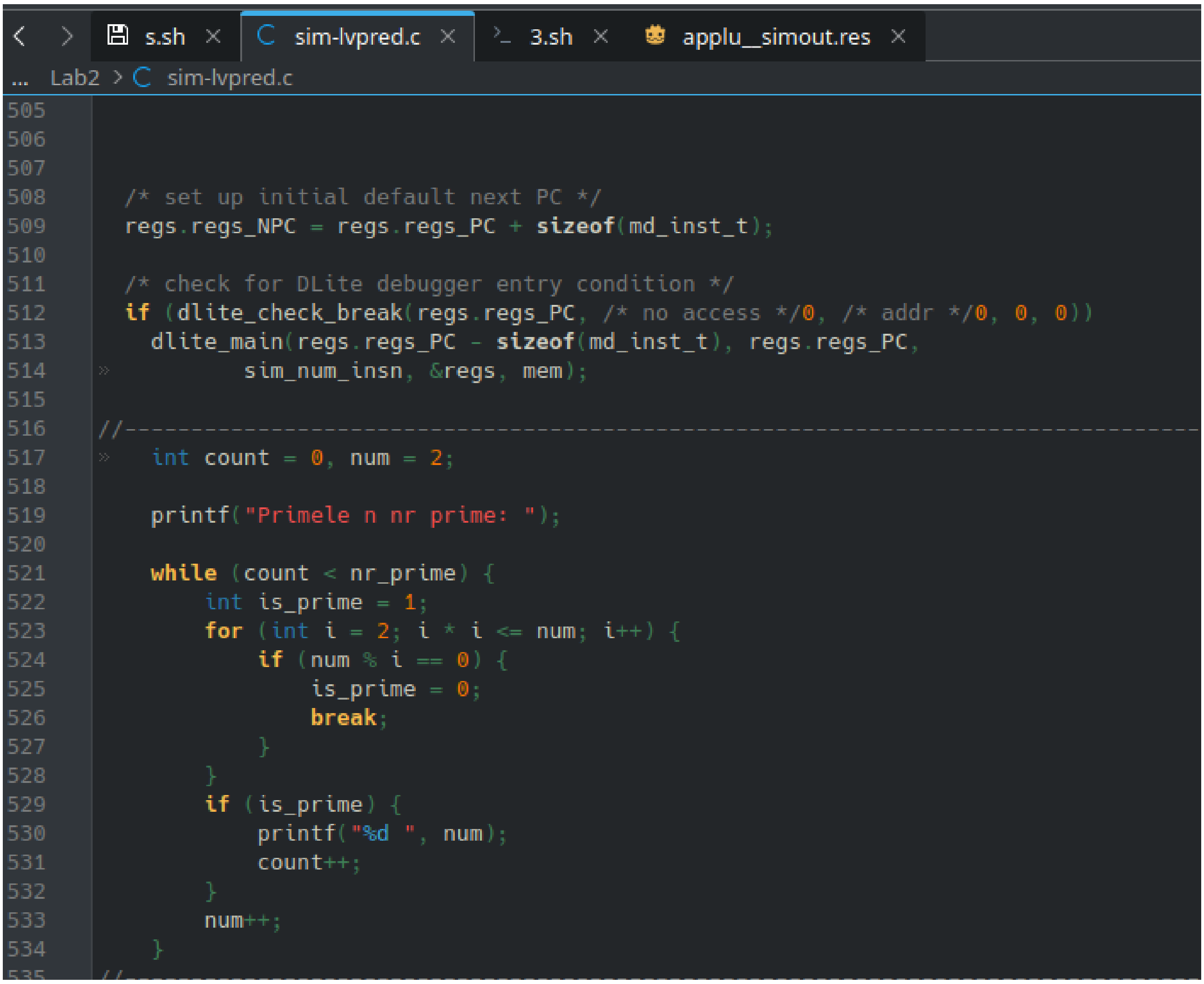
2. (-history n) cu acesta vom calcula si afisa primele n numere prime, unde n reprezinta paramentrul dat de utilizator. (done)



```
                );

  /* instruction limit */
  opt_reg_uint(odb, "-max:inst", "maximum number of inst's to exe
           &max_insts, /* default */0,
           /* print */TRUE, /* format */NULL);

 opt_reg_uint(odb, "-contor:LD", "contor LD",
           &opt_load, /* default */0,
           /* print */TRUE, /* format */NULL);

 opt_reg_uint(odb, "-history", "history",
           &nr_prime, /* default */0,
           /* print */TRUE, /* format */NULL);
```



```
 opt_reg_uint(odb, "-contor:LD", "contor LD",
           &opt_load, /* default */0,
           /* print */TRUE, /* format */NULL);

 opt_reg_uint(odb, "-history", "history",
           &nr_prime, /* default */0,
           /* print */TRUE, /* format */NULL);

```

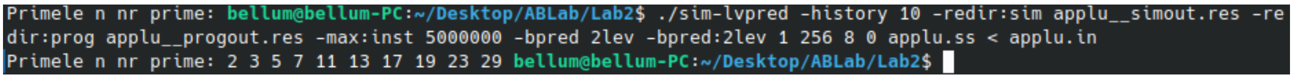


```c



  /* set up initial default next PC */
  regs.regs_NPC = regs.regs_PC + sizeof(md_inst_t);

  /* check for DLite debugger entry condition */
  if (dlite_check_break(regs.regs_PC, /* no access */0, /* addr */0, 0, 0))
    dlite_main(regs.regs_PC - sizeof(md_inst_t), regs.regs_PC,
           sim_num_insn, &regs, mem);

//--------------------------------------------------------------------------------
    int count = 0, num = 2;

    printf("Primele n nr prime: ");

    while (count < nr_prime) {
        int is_prime = 1;
        for (int i = 2; i * i <= num; i++) {
            if (num % i == 0) {
                is_prime = 0;
                break;
            }
        }
        if (is_prime) {
            printf("%d ", num);
            count++;
        }
        num++;
    }
//--
```

Comanda data a fost:

./sim-lvpred -history 10 -redir:sim applu__simout.res -redir:prog applu__progout.res -max:inst 5000000 -bpred 2lev -bpred:2lev 1 256 8 0 applu.ss < applu.in

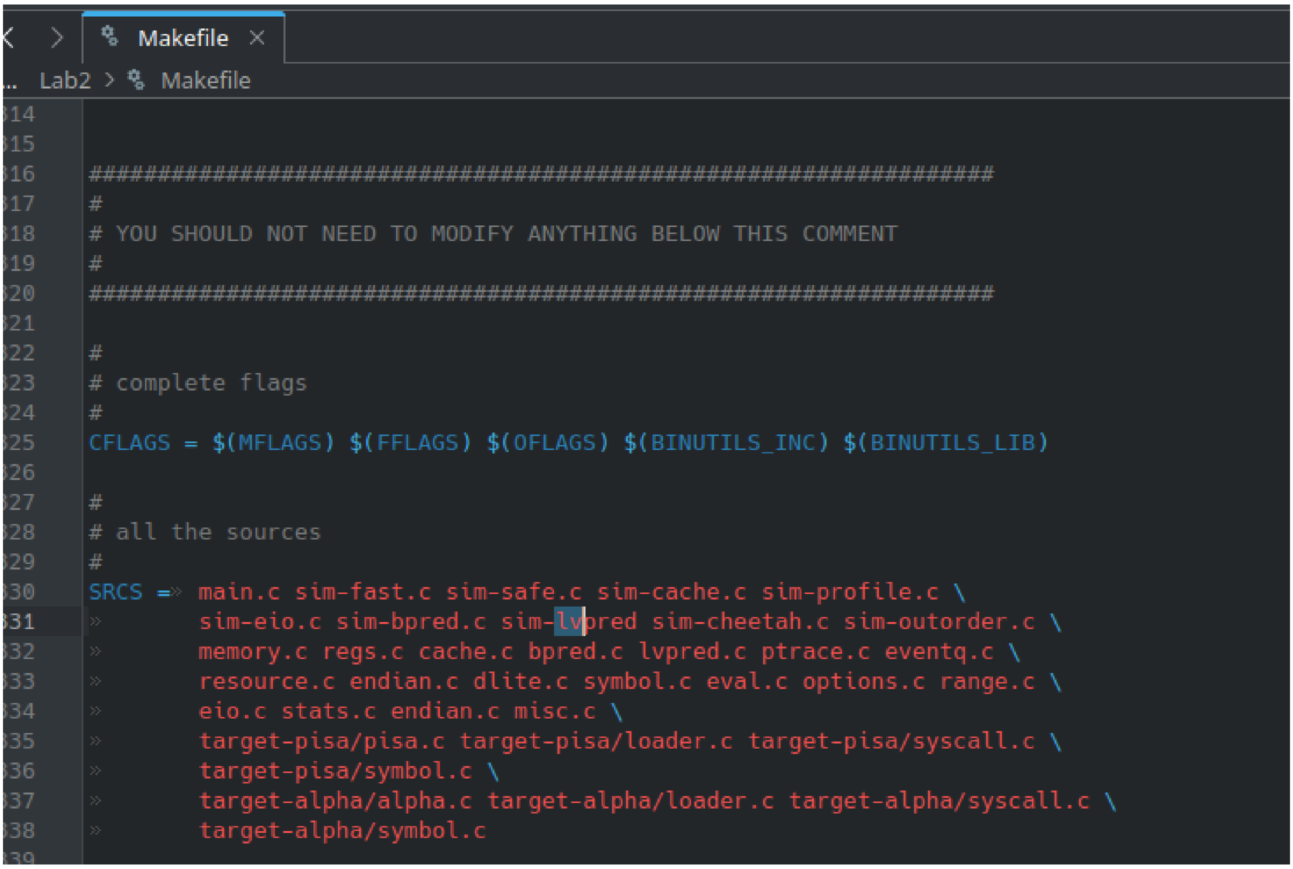
Iar rezultatul:



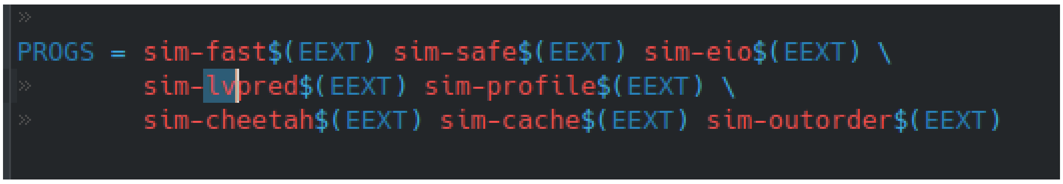
```
Primele n nr prime: bellum@bellum-PC:~/Desktop/ABLab/Lab2$ ./sim-lvpred -history 10 -redir:sim applu__simout.res -re
dir:prog applu__progout.res -max:inst 5000000 -bpred 2lev -bpred:2lev 1 256 8 0 applu.ss < applu.in
Primele n nr prime: 2 3 5 7 11 13 17 19 23 29 bellum@bellum-PC:~/Desktop/ABLab/Lab2$ 
```

3. Adaugarea unor optiuni de masurare a gradelor de localitate din benchmark-uri. (not done yet...)

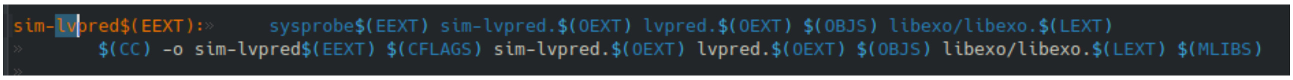
De asemenea e de mentionat ca makefile-ul necesar compilari a fost de asemenea modificat ca sa corespunda compilarii lui lv-pred



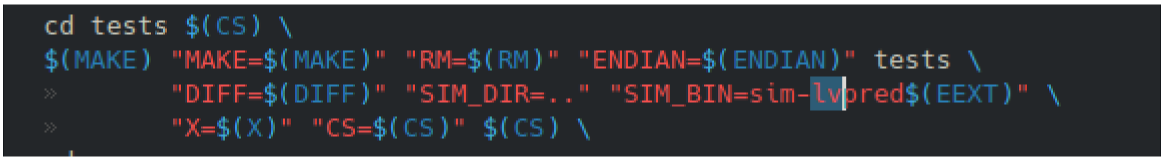
```
###################################################################
#
# YOU SHOULD NOT NEED TO MODIFY ANYTHING BELOW THIS COMMENT
#
###################################################################

#
# complete flags
#
CFLAGS = $(MFLAGS) $(FFLAGS) $(OFLAGS) $(BINUTILS_INC) $(BINUTILS_LIB)

#
# all the sources
#
SRCS =	main.c sim-fast.c sim-safe.c sim-cache.c sim-profile.c \
	sim-eio.c sim-bpred.c sim-lvpred sim-cheetah.c sim-outorder.c \
	memory.c regs.c cache.c bpred.c lvpred.c ptrace.c eventq.c \
	resource.c endian.c dlite.c symbol.c eval.c options.c range.c \
	eio.c stats.c endian.c misc.c \
	target-pisa/pisa.c target-pisa/loader.c target-pisa/syscall.c \
	target-pisa/symbol.c \
	target-alpha/alpha.c target-alpha/loader.c target-alpha/syscall.c \
	target-alpha/symbol.c
```



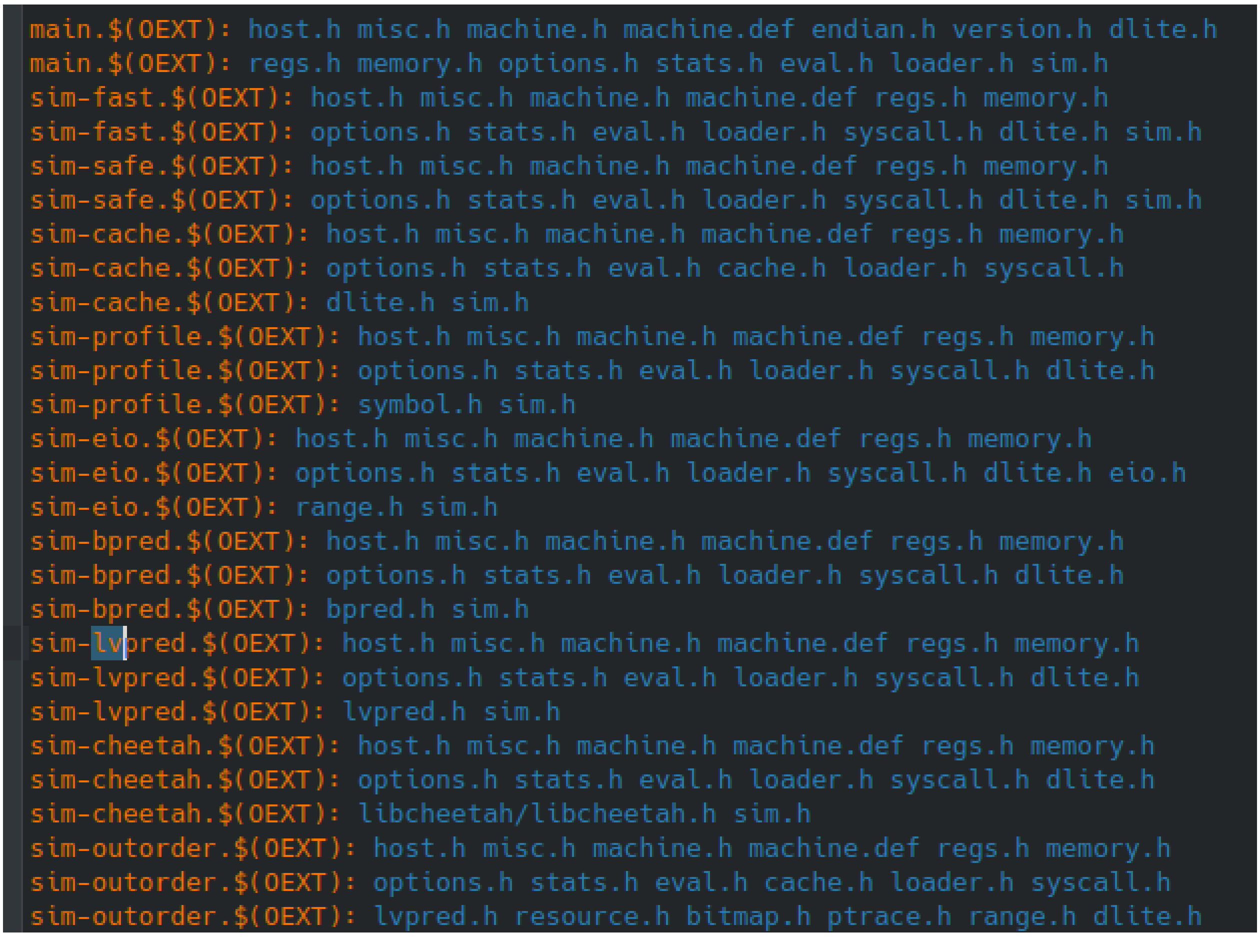
```
PROGS = sim-fast$(EEXT) sim-safe$(EEXT) sim-eio$(EEXT) \
	sim-lvpred$(EEXT) sim-profile$(EEXT) \
	sim-cheetah$(EEXT) sim-cache$(EEXT) sim-outorder$(EEXT)
```



```
sim-lvpred$(EEXT):	sysprobe$(EEXT) sim-lvpred.$(OEXT) lvpred.$(OEXT) $(OBJS) libexo/libexo.$(LEXT)
	$(CC) -o sim-lvpred$(EEXT) $(CFLAGS) sim-lvpred.$(OEXT) lvpred.$(OEXT) $(OBJS) libexo/libexo.$(LEXT) $(MLIBS)
```



```
cd tests $(CS) \
$(MAKE) "MAKE=$(MAKE)" "RM=$(RM)" "ENDIAN=$(ENDIAN)" tests \
	"DIFF=$(DIFF)" "SIM_DIR=.." "SIM_BIN=sim-lvpred$(EEXT)" \
	"X=$(X)" "CS=$(CS)" $(CS) \
```



```
main.$(OEXT): host.h misc.h machine.h machine.def endian.h version.h dlite.h
main.$(OEXT): regs.h memory.h options.h stats.h eval.h loader.h sim.h
sim-fast.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-fast.$(OEXT): options.h stats.h eval.h loader.h syscall.h dlite.h sim.h
sim-safe.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-safe.$(OEXT): options.h stats.h eval.h loader.h syscall.h dlite.h sim.h
sim-cache.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-cache.$(OEXT): options.h stats.h eval.h cache.h loader.h syscall.h
sim-cache.$(OEXT): dlite.h sim.h
sim-profile.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-profile.$(OEXT): options.h stats.h eval.h loader.h syscall.h dlite.h
sim-profile.$(OEXT): symbol.h sim.h
sim-eio.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-eio.$(OEXT): options.h stats.h eval.h loader.h syscall.h dlite.h eio.h
sim-eio.$(OEXT): range.h sim.h
sim-bpred.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-bpred.$(OEXT): options.h stats.h eval.h loader.h syscall.h dlite.h
sim-bpred.$(OEXT): bpred.h sim.h
sim-lvpred.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-lvpred.$(OEXT): options.h stats.h eval.h loader.h syscall.h dlite.h
sim-lvpred.$(OEXT): lvpred.h sim.h
sim-cheetah.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-cheetah.$(OEXT): options.h stats.h eval.h loader.h syscall.h dlite.h
sim-cheetah.$(OEXT): libcheetah/libcheetah.h sim.h
sim-outorder.$(OEXT): host.h misc.h machine.h machine.def regs.h memory.h
sim-outorder.$(OEXT): options.h stats.h eval.h cache.h loader.h syscall.h
sim-outorder.$(OEXT): lvpred.h resource.h bitmap.h ptrace.h range.h dlite.h
```