دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

پروژه اول درس ریزپردازنده و زبان اسمبلی

عنوان پروژه:

بهینه‌سازی عملیات شبکه‌های عصبی کانولوشنی با استفاده از دستورالعمل‌های سفارشی در معماری RISC-V

نگارش:

لیلاالسادات محسنی

شماره دانشجویی : ۴۰۱۳۱۰۴۴

استاد درس :

دکتر حامد فربه

تاریخ تحویل :

۱۴۰۴/۰۵/۰۷

فهرست مطالب

عملیات پیاده سازی شده در کل پروژه : ضرب دو ماتریس(mac)

بخش اول : پیاده‌سازی با RV32IM و اجرا در محیط شبیه‌ساز

توضیحات کد اسمبلی :

در ابتدا ماکرو Find_element را برای دسترسی به یک عنصر خاص از ماتریس دوبعدی در حافظه تعریف می‌کنیم. از آنجا که در حافظه ابتدا تعداد سطر و سپس تعداد ستون ذخیره شده‌اند، ماکرو از روی این دو مقدار عبور می‌کند تا به داده‌های اصلی برسد. با استفاده از فرمول (row - 1) * num_cols + (col−1) موقعیت عنصر موردنظر را در آرایه یک‌بعدی محاسبه می‌کند و آن را در ۴ (اندازه هر عنصر) ضرب می‌کند تا offsetبایتی به‌دست آید. در آخر با اضافه کردن این offset به آدرس ماتریس، به آدرس عنصر می‌رسد و مقدار آن را در رجیستر موردنظر ذخیره می‌کند.

در قسمت data دو ماتریس A و B را تعریف می‌کنیم سپس در بخش start مقداردهی اولیه به پوینترها می‌کنیم و s6 هم برای نتیجه استفاده می‌شود. در بخش check_dimensions ابعاد ماتریس‌هارا چک می‌کنیم که برایشان ضرب قابل تعریف هست یا نه. یعنی خانه دوم پوینترمربوط به ماتریس اول را در s4 لود می‌کنیم و خانه اول پوینترمربوط به ماتریس دوم را در s7 لود می‌کنیم و بعد مقایسه می‌کنیم اگر برابر نبودند به بخش end می‌رویم و برنامه تمام می‌شود. در بخش save_result_dimensions ماتریس نتیجه را ذخیره می‌کنیم یعنی مقدار موجود در s3 که تعداد ردیف‌های ماتریس اول است در خانه اول s6 و برای رفتن به خانه‌ی دیگر از دستور ADDI استفاده می‌کنیم که ۴ بایت جلو می‌رود.این کار را برای مشخص کردن تعداد ستون‌های ماتریس نتیجه نیز انجام می‌دهیم.

حالا سه تا لیبل for تعریف می‌کنیم در اولی مقدار اولیه را یک ست می‌کنیم و شرط را این قرار می‌دهیم که اگر مقدارش بزرگتر از s3 یعنی تعداد سطرهای ماتریس اول شد به انتهای لیبل برود. در حلقه میانی مقدار اولیه را مجددا یک ست می‌کنیم و شرط را این قرار می‌دهیم که اگر مقدارش بزرگتر از s5 یعنی تعداد ستون های ماتریس دوم شد به انتهای لیبل برود. حالا حلقه درونی را تعریف می‌کنیم ابتدا مقدار اولیه را مثل دو حلقه قبلی یک ست می‌کنیم و بعد s9 را که برای مقدار هر درایه است صفر مقداردهی می‌کنیم. اگر مقدار t5 از s4 که تعدادستون ماتریس اول(تعداد سطر ماتریس دوم) است بیشتر شد به انتهای حلقه می‌رود. سپس مقدار t6 و s7 را لود می‌کنیم که مقدار t6 برابر A[i][k] و مقدار s7 برابر B[k][j] است در نهایت از عمل ضرب استفاده می‌کنیم و نتیجه را در s9 ذخیره می‌کنیم و بعد مقدار را با s9 جمع و آپدیت می‌کنیم و حلقه را جلو می‌بریم. مقدار نتیجه را در s6[0] ذخیره می‌کنیم و به خانه بعدی ماتریس می‌رویم.حلقه میانی را جلو می‌بریم و بعد اتمام حلقه‌ها را داریم.

کد :

Editor (Ctrl-E)

Compile and Load (F5) Language: RV32 P1_micro (1).s [changed since save] [changed since compile]

```
# Load a matrix element at [row][col] into el
.macro Find_element, Matrix, X, row, col
    ADDI \Matrix, \Matrix, 4         # Skip row count
    LW t1, 0(\Matrix)              # t1 ← number of columns
    ADDI \Matrix, \Matrix, 4         # Skip column count
    ADDI \row, \row, -1
    MUL t0, \row, t1           # t0 = row * num_cols
    ADD t2, t0, \col
    ADDI t2, t2, -1
    LI t0, 4
    MUL t2, t2, t0             # offset = (row * col + col - 1) * 4
    ADD \Matrix, \Matrix, t2
    LW \X, 0(\Matrix)
    ADDI \Matrix, \Matrix, -8
    SUB \Matrix, \Matrix, t2
    ADDI \row, \row, 1
.endm
_start:
    LA s1, first_matrix
    LA s2, second_matrix
    LA s6, result
check_dimensions:
    LW s3, 0(s1)
    LW s4, 4(s1)
    LW s7, 0(s2)
    LW s5, 4(s2)

    BNE s4, s7, end
    LI s7, 0
save_result_dimensions:
    SW s3, 0(s6)
    ADDI s6, s6, 4
    SW s5, 0(s6)
    ADDI s6, s6, 4

```

Editor (Ctrl-E) Disassembly (Ctrl-D) Memory (Ctrl-M)

Compile and Load (F5) Language: RV32 P1_micro (1).s [changed since save] [changed since compile]

```
zarb_matrix:
    LI t3, 1                   # t3 = row_i
loop_row:
    BGT t3, s3, end_loop_row
    LI t4, 1                   # t4 = col_j
loop_col:
    BGT t4, s5, end_loop_col
    LI t5, 1                   # t5 = k
    LI s9, 0                   # s9 = accumulator for result[t3][t4]
loop_k:
    BGT t5, s4, end_loop_k
    # t6 = A[t3][k]
    Find_element s1, t6, t3, t5
    # s7 = B[k][t4]
    Find_element s2, s7, t5, t4
    MUL s8, t6, s7
    ADD s9, s9, s8
    ADDI t5, t5, 1
    J loop_k
end_loop_k:
    SW s9, 0(s6)               # store result[t3][t4]
    ADDI s6, s6, 4
    ADDI t4, t4, 1
    J loop_col
end_loop_col:
    ADDI t3, t3, 1
    J loop_row
end_loop_row:
    LI t0, 0xffffffff          # marker
    SW t0, 0(s6)
end:
    J end


.data
```

Editor (Ctrl-E) Disassembly (Ctrl-D) Memory (Ctrl-M)

ورودی:

```
.data

first_matrix:
.word 2 ,3
.word 1, 2,3
.word 3 ,2, 1


second_matrix:
.word 3,2
.word 1, 2
.word 2 ,1
.word 1, 2

.word 0xffffffff              # marker before result
result:
.space 64
```

خروجی:

Go to address, label, or register: 0x000000dc Refresh

Address	Memory contents and ASCII				
00000020	2963	20652067	4918035	22749219	••• # ;• ••K• # [•
00000030	4918035	1052179	197773411	1052307	••K• ••• c••• •••
00000040	198885475	1052435	3219	166347363	c••• ••• ••• cB••
00000050	4490387	303875	4490387	4294839827	••D• ••• ••D• •••
00000060	40764083	31622067	4294149011	4194963	••n• ••• ••• ••@•
00000070	39027635	7636147	307075	4286874771	••S• ••t• ••• •••
00000080	1081377971	1969683	4786451	598787	••t@ ••• ••I• •#••
00000090	4786451	4294905619	40829619	30573491	••I• ••• ••o• •••
000000a0	4294149011	4194963	39027635	7932211	••• ••@• ••S• 3•y•
000000b0	600963	4287170835	1081674035	2035475	•+•• ••• 3•y@ •••
000000c0	58690611	25988275	2035475	4162842735	3••• ••• ••• o•••
000000d0	26943523	4918035	2002579	4133482607	# •• ••K• ••• o•_•
000000e0	1969683	4116705391	4293919379	5972003	••• o•_• ••• # [•
000000f0	111	2	3	1	o••• ••• ••• •••
00000100	2	3	3	2	•••• •••• •••• ••••
00000110	1	3	2	1	•••• •••• •••• ••••
00000120	2	2	1	1	•••• •••• •••• ••••
00000130	2	4294967295	2	2	•••• •••• •••• ••••
00000140	8	10	8	10	•••• •••• •••• ••••
00000150	4294967295	0	0	0	•••• •••• •••• ••••
00000160	0	0	0	0	•••• •••• •••• ••••
00000170	0	0	4294967295	0	•••• •••• •••• ••••
00000180	0	0	0	0	•••• •••• •••• ••••
00000190	0	0	2863311530	2863311530	•••• •••• •••• ••••
000001a0	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
000001b0	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
000001c0	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
000001d0	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
000001e0	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
000001f0	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
00000200	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
00000210	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
00000220	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
00000230	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
00000240	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
00000250	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
00000260	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••
00000270	2863311530	2863311530	2863311530	2863311530	•••• •••• •••• ••••

بخش دوم: اضافه کردن دستورالعمل‌های سفارشی و پیاده‌سازی

توضیحات نصب :

در این پروژه من از اوبونتو استفاده کرده‌ام. ابتدا فولدرهای موردنیاز را کلون کرده و بعد بیلد را انجام داده‌ام.
در ادامه برای آماده سازی فولدرها و فایل‌ها طبق سایت گفته شده در پروژه مرحله به مرحله پیش رفته‌ام.

(https://pcotret.gitlab.io/riscv-custom/sw_toolchain.html)

نصب tool-chain و مشاهده ورژن :

```
leila@DESKTOP-DE34NSM:~$ /opt/riscv_custom/bin/./riscv64-unknown-elf-gcc --version
riscv64-unknown-elf-gcc (g1b306039a) 15.1.0
Copyright (C) 2025 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

اضافه کردن دستورسفارشی به کراس کامپایلر:

در مسیر زیر و درفایل مربوطه دستور mac را اضافه می‌کنیم.

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-opcodes/extensions$ nano rv_i
```

```
srl      rd rs1 rs2 31..25=0  14..12=5 6..2=0x0C 1..0=3
sra      rd rs1 rs2 31..25=32 14..12=5 6..2=0x0C 1..0=3
or       rd rs1 rs2 31..25=0  14..12=6 6..2=0x0C 1..0=3
and      rd rs1 rs2 31..25=0  14..12=7 6..2=0x0C 1..0=3
mac      rd rs1 rs2 31..25=1 14..12=0 6..2=0x0B 1..0=3
```

همچنین در مسیر زیر match و mask را برای دستورmac ست می‌کنیم و قرار می‌دهیم. درآخر دستور را تعریف می‌کنیم.

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ cd binutils
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/binutils$ cd include
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/binutils/include$ cd opcode
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/binutils/include/opcode$ nano riscv-opc.h
```

```
#ifndef RISCV_ENCODING_H
#define RISCV_ENCODING_H
/* MY instructions */
#define MATCH_MAC 0x200000b
#define MASK_MAC  0xfe00707f
/* end */
```

```
#ifdef DECLARE_INSN
DECLARE_INSN(mac, MATCH_MAC, MASK_MAC)
```

همچنین در مسیر زیر نیز دستور mac را تعریف می‌کنیم.

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/binutils/include/opcode$ cd ..
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/binutils/include$ cd ..
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/binutils$ cd opcodes
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/binutils/opcodes$ nano riscv-opc.c
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/binutils/opcodes$
```

```
/* Basic RVI instructions and aliases.  */
{"mac",         0, INSN_CLASS_I, "d,s,t",     MATCH_MAC, MASK_MAC, match_opcode, 0 },
{"unimp",       0, INSN_CLASS_C, "",          0, 0xffffU, match_opcode, INSN_ALIAS },
{"unimp",       0, INSN_CLASS_I, "",          MATCH_CSRRW|(CSR_CYCLE << OP_SH_CSR), 0xffffffffU,  match_opcode, 0 }, /* csrw cycle, x0
{"ebreak",      0, INSN_CLASS_C, "",          MATCH_C_EBREAK, MASK_C_EBREAK, match_opcode, INSN_ALIAS },
```

در آخر با زدن دستور clean و make مجددا کلون می‌کنیم.

اضافه کردن دستور سفارشی به spike :

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ export RISCV=/opt/riscv_custom
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ export PATH=$RISCV/bin:$PATH
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ git clone https://github.com/riscv-software-src/riscv-isa-sim
fatal: destination path 'riscv-isa-sim' already exists and is not an empty directory.
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ cd riscv-isa-sim
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim$ mkdir build
mkdir: cannot create directory ‘build’: File exists
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim$ cd build
```

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/build$ ../configure --prefix=$RISCV
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking for g++... g++
checking whether the compiler supports GNU C++... yes
checking whether g++ accepts -g... yes
checking for g++ option to enable C++11 features... none needed
checking for ar... ar
checking for ranlib... ranlib
checking for dtc... /usr/bin/dtc
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
```

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/build$ make -j$(nproc)
make: Circular libriscv.so <- libriscv.so dependency dropped.
make: Circular libcustomext.so <- libcustomext.so dependency dropped.
make: Circular libsoftfloat.so <- libsoftfloat.so dependency dropped.
make: Nothing to be done for 'default'.
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/build$ sudo make install
[sudo] password for leila:
../scripts/mk-install-dirs.sh /opt/riscv_custom/include
for file in fesvr/byteorder.h fesvr/elf.h fesvr/elfloader.h fesvr/htif.h fesvr/dtm.h fesvr/memif.h fesvr/syscall.h fesvr/context.h fesvr/htif_pthread.h fesvr/htif_hexwriter.h fesvr/option_parser.h fesvr/term.h f
esvr/device.h fesvr/rfb.h fesvr/tsi.h riscv/abstract_device.h riscv/abstract_interrupt_controller.h riscv/cachesim.h riscv/cfg.h riscv/common.h riscv/csrs.h riscv/debug_defines.h riscv/debug_module.h riscv/debug
_rom_defines.h riscv/decode.h riscv/devices.h riscv/disasm.h riscv/dts.h riscv/encoding.h riscv/entropy_source.h riscv/extension.h riscv/isa_parser.h riscv/jtag_dtm.h riscv/log_file.h riscv/memtracer.h riscv/mmu
.h riscv/platform.h riscv/processor.h riscv/remote_bitbang.h riscv/rocc.h riscv/sim.h riscv/simif.h riscv/trap.h riscv/triggers.h riscv/vector_unit.h fdt/libfdt.h fdt/fdt.h fdt/libfdt_env.h softfloat/softfloat.h
 softfloat/softfloat_types.h; \
  do \
        ../scripts/mk-install-dirs.sh /opt/riscv_custom/include/`dirname $file`; \
    /usr/bin/install -c -m 644 ../$file /opt/riscv_custom/include/`dirname $file`; \
  done
../scripts/mk-install-dirs.sh /opt/riscv_custom/include
for dir in fesvr       ; \
  do \
        ../scripts/mk-install-dirs.sh /opt/riscv_custom/include/$dir; \
    /usr/bin/install -c -m 644 config.h /opt/riscv_custom/include/$dir; \
  done
make: Circular libriscv.so <- libriscv.so dependency dropped.
make: Circular libcustomext.so <- libcustomext.so dependency dropped.
make: Circular libsoftfloat.so <- libsoftfloat.so dependency dropped.
../scripts/mk-install-dirs.sh /opt/riscv_custom/lib
for file in libfesvr.a libriscv.so libdisasm.a libcustomext.so libsoftfloat.so; \
  do \
    /usr/bin/install -c -m 644 $file /opt/riscv_custom/lib; \
  done
../scripts/mk-install-dirs.sh /opt/riscv_custom/bin
for file in elf2hex spike spike-log-parser xspike termios-xspike spike-dasm; \
  do \
    /usr/bin/install -c -m 755 $file /opt/riscv_custom/bin; \
  done
../scripts/mk-install-dirs.sh /opt/riscv_custom/lib/pkgconfig/
for file in riscv-fesvr.pc riscv-riscv.pc riscv-disasm.pc; \
  do \
    /usr/bin/install -c -m 644 $file /opt/riscv_custom/lib/pkgconfig/; \
  done
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/build$
```

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/build$ cd ..
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim$ cd ..
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ git clone https://github.com/riscv-software-src/riscv-pk
fatal: destination path 'riscv-pk' already exists and is not an empty directory.
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ mkdir build
mkdir: cannot create directory ‘build’: File exists
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ cd build
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/build$ ../configure --prefix=$RISCV --host=riscv64-unknown-e
checking for riscv64-unknown-elf-gcc... riscv64-unknown-elf-gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... yes
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether riscv64-unknown-elf-gcc accepts -g... yes
checking for riscv64-unknown-elf-gcc option to enable C11 features... none needed
checking for grep that handles long lines and -e... /usr/bin/grep
checking for fgrep... /usr/bin/grep -F
checking for grep that handles long lines and -e... (cached) /usr/bin/grep
checking for bash... /bin/bash
checking for __gmpz_init in -lgmp... no
checking for mpfr_init in -lmpfr... no
checking for mpc_init2 in -lmpc... no
checking for curl... /usr/bin/curl
checking for wget... /usr/bin/wget
checking for ftp... no
configure: creating ./config.status
config.status: creating Makefile
config.status: creating scripts/wrapper/awk/awk
config.status: creating scripts/wrapper/sed/sed
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/build$ make -j$(nproc)
make: Nothing to be done for 'all'.
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/build$ sudo make install
make: Nothing to be done for 'install'.
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/build$ export PATH=$RISCV/riscv64-unknown-elf/bin:$PATH
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/build$
```

اضافه کردن دستور سفارشی به شبیه ساز:

دستور mac را با آدرس زیر در فایل مربوطه تعریف می‌کنیم.

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ cd riscv-isa-sim
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim$ cd riscv
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/riscv$ cd insns
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/riscv/insns$ nano mac.h
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/riscv/insns$
```

```
  GNU nano 7.2
WRITE_RD(sext_xlen(RD + (RS1 * RS2)));
```

حالا در این مرحله mask و match دستور را به فایل encoding اضافه می‌کنیم.

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/riscv/insns$  cd ..
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/riscv$ nano encoding.h
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/riscv$
```

```
/* SPDX-License-Identifier: BSD-3-Clause */

/* Copyright (c) 2023 RISC-V International */

/*
 * This file is auto-generated by running 'make' in
 * https://github.com/riscv/riscv-opcodes (8899b32)
 */

#ifndef RISCV_CSR_ENCODING_H
#define RISCV_CSR_ENCODING_H
/* my instruction*/
#define MATCH_MAC 0x200000b
#define  MASK_MAC 0xfe00707f
/* end */
```

```
#ifdef DECLARE_INSN
DECLARE_INSN(mac, MATCH_MAC, MASK_MAC)
DECLARE_INSN(add, MATCH_ADD, MASK_ADD)
DECLARE_INSN(add_uw, MATCH_ADD_UW, MASK_ADD_UW)
DECLARE_INSN(addi, MATCH_ADDI, MASK_ADDI)
```

سپس به آدرس زیر دستور mac را به سایر riscv-isns-ext-i اضافه می‌کنیم.

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim$ cd riscv
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/riscv$ nano riscv.mk.in
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/riscv$
```

```
riscv_insn_ext_i = \
        add \
        mac \
        addi \
        addiw \
        addw \
        and \
        andi \
        auipc \
```

و در این مسیر نیز همین‌جور که در داک پروژه گفته شده بود دستور mac که از نوع Rtype است را اضافه می‌کنیم.

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim$ cd disasm
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/disasm$ nano disasm.cc
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain/riscv-isa-sim/disasm$
```

```
DEFINE_RTYPE(add);
DEFINE_RTYPE(mac);
DEFINE_RTYPE(sub);
DEFINE_RTYPE(sll);
DEFINE_RTYPE(slt);
```

تست برنامه ساده با دستور mac :

در این کد که ترکیبی از زبان C و inline Assembly برای معماری RISC-V است از دستور سفارشی mac که اضافه کرده بودیم استفاده می‌کنیم.این دستور در پروژه به صورت زیر تعریف شده است:

$$\text{mac rd, rs1, rs2} \quad \rightarrow \quad \text{rd} = \text{rd} + (\text{rs1} * \text{rs2})$$

در ابتدای کد سه متغیر a,b,c را تعریف و مقداردهی می‌کنیم.در بخش بعدی از mac که تعریف کرده‌ایم استفاده می‌کنیم در خط اول تعریف متغیرهای ورودی (op1 , op2) و متغیر خروجی(res) را داریم. در

خط دوم نشان می‌دهیم که c در نهایت در res قرار می‌گیرد و در خط سوم متغیر a جایگزین op1 و متغیر b جایگزین op2 می‌شود. در نهایت نتیجه نیز چاپ می‌شود.در کد ما عملیات به این صورت انجام می‌شود:

$$c = c + a * b \rightarrow 20 + 5 * 6 = 50$$

```c
#include <stdio.h>

int main() {
    int a = 6;
    int b = 5;
    int c = 20;

    asm volatile(
        "mac %[res], %[op1], %[op2]\n\t"
        : [res] "+r" (c)
        : [op1] "r" (a), [op2] "r" (b)
    );

    printf("Result of mac: %d\n", c);
    return 0;
}
```

نمونه ورودی و خروجی :

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ /opt/riscv_custom/bin/riscv64-unknown-elf-gcc -o mac_test mac_test.c
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ file mac_test
mac_test: ELF 64-bit LSB executable, UCB RISC-V, RVC, double-float ABI, version 1 (SYSV), statically linked, with debug_info, not stripped
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ spike /opt/riscv_custom/riscv64-unknown-elf/bin/pk ./mac_test
Result of mac: 50
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$
```

تست برنامه ضرب ماتریس با دستور mac:

در این کد ابتدا ماتریس های ورودی A و B و ماتریس نهایی ضرب یعنی C را تعریف می‌کنیم. سپس ابعاد هردوماتریس را از کاربر به عنوان ورودی دریافت می‌کنیم.اگر تعداد ستون های ماتریس اول با تعداد سطرهای ماتریس دوم برابر نباشد ضرب امکان پذیر نیست و پیغام مناسب چاپ می‌شود و برنامه پایان می یابد.در صورت برقراری شرط درایه‌های هردو ماتریس ورودی گرفته می‌شود. حالا سطر به سطر در ماتریس A جلو رفته برای هر سطر آن تمام ستون های B را پیمایش می‌کنیم تا مقدار هر درایه C محاسبه شود.حالا برای استفاده از دستور mac از A[i][k] به عنوان a و از B[k][j] به عنوان b استفاده می‌کنیم.به جای اینکه عبارت $sum += sum + a * b$ را بنویسیم از این دستور استفاده کرده‌ایم. در نهایت مقدار sum به عنوان C[i][j] قرار می‌گیرد. در نهایت ماتریسی که تعداد سطر به اندازه تعداد سطرهای A و تعداد ستون به اندازه تعداد ستون‌های B دارد را به عنوان خروجی چاپ می‌کنیم.

```c
#include <stdio.h>

#define MAX 10

int main() {
    int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX] = {0};
    int rowA, colA, rowB, colB;

    printf("Enter rows and columns for matrix A: ");
    scanf("%d %d", &rowA, &colA);

    printf("Enter rows and columns for matrix B: ");
    scanf("%d %d", &rowB, &colB);

    if (colA != rowB) {
        printf("Matrix multiplication not possible.\n");
        return 1;
    }

    printf("Enter elements of matrix A:\n");
    for (int i = 0; i < rowA; i++) {
        for (int j = 0; j < colA; j++) {
            scanf("%d", &A[i][j]);
        }
    }

    printf("Enter elements of matrix B:\n");
    for (int i = 0; i < rowB; i++) {
        for (int j = 0; j < colB; j++) {
            scanf("%d", &B[i][j]);
        }
    }
```

```c
    for (int i = 0; i < rowA; i++) {
        for (int j = 0; j < colB; j++) {
            int sum = 0;
            for (int k = 0; k < colA; k++) {
                int a = A[i][k], b = B[k][j];
                asm volatile(
                    "mac %[res], %[op1], %[op2]\n\t"
                    : [res] "+r" (sum)
                    : [op1] "r" (a), [op2] "r" (b)
                );
            }
            C[i][j] = sum;
        }
    }

    printf("Resulting matrix C:\n");
    for (int i = 0; i < rowA; i++) {
        for (int j = 0; j < colB; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

حالات مختلف ورودی و خروجی :

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ /opt/riscv_custom/bin/riscv64-unknown-elf-gcc -o zarb_matrix zarb_matrix.c
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ file zarb_matrix
zarb_matrix: ELF 64-bit LSB executable, UCB RISC-V, RVC, double-float ABI, version 1 (SYSV), statically linked, with debug_info, not stripped
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ spike /opt/riscv_custom/riscv64-unknown-elf/bin/pk ./zarb_matrix
Enter rows and columns for matrix A: 1 2
Enter rows and columns for matrix B: 1 3
Matrix multiplication not possible.
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$
```

```
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$ spike /opt/riscv_custom/riscv64-unknown-elf/bin/pk ./zarb_matrix
Enter rows and columns for matrix A: 2 3
Enter rows and columns for matrix B: 3 2
Enter elements of matrix A:
1 2 3
3 2 1
Enter elements of matrix B:
1 2
2 1
1 2
Resulting matrix C:
8 10
8 10
leila@DESKTOP-DE34NSM:~/riscv-gnu-toolchain$
```