LAB 1

1 gem5的安装和测试

编译指定架构的处理器：

```
$ scons build/ARM/gem5.debug -j 4
```

编译`build/ARM/gem5.debug`后，使用如下指令进行测试：

```
$ build/ARM/gem5.debug configs/example/se.py -c tests/test-
progs/hello/bin/arm/linux/hello
```

输出结果如下：

```
[gem5-stable] build/ARM/gem5.debug configs/example/se.py -c tests/test-progs/hello/bin/arm/linux/hello
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 20.1.0.4
gem5 compiled Mar 11 2023 23:33:38
gem5 started Mar 12 2023 09:29:52
gem5 executing on Ther, pid 1706
command line: build/ARM/gem5.debug configs/example/se.py -c tests/test-progs/hello/bin/arm/linux/hello

warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
warn: membus.slave is deprecated. `slave` is now called `cpu_side_ports`
warn: membus.master is deprecated. `master` is now called `mem_side_ports`
Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and pdf.
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
Hello world!
Exiting @ tick 2916500 because exiting with last active thread context
Simulated exit code not 0! Exit code is 13
```

2 交叉编译

2.1 交叉编译器的安装使用

安装ARM指令集交叉编译器`gcc-aarch64-linux-gnu`：

```
$ sudo apt install gcc-aarch64-linux-gnu
```

对八皇后程序`queens.c`进行交叉编译：

```
$ aarch64-linux-gnu-gcc -o labs/lab1/queens labs/lab1/queens.c --static
```

根据所提供的参数，在ARM架构上对可执行程序进行仿真：

```
$ build/ARM/gem5.debug configs/example/se.py -c labs/lab1/queens --cpu-
type=MinorCPU --caches --l2cache --l1d_size=8kB --l1i_size=8kB --l1d_assoc=4 --
l1i_assoc=4 --l2_size=1MB --l2_assoc=8 --cacheline_size=64  --cpu-clock=1GHz --
num-cpus=1
```

仿真结果如下：

```
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
info: Increasing stack size by one page.
warn:   instruction 'bti' unimplemented
warn: ignoring syscall mprotect(...)
8 queens on a 8x8 board...
 Q - - - - - - -
 - - - - Q - - -
 - - - - - - - Q
 - - - - - Q - -
 - - Q - - - - -
 - - - - - - Q -
 - Q - - - - - -
 - - - Q - - - -
Exiting @ tick 85241000 because exiting with last active thread context
```

2.2 如何得到其他数量的皇后对应的解？如何得到指定皇后数对应的解的总个数？

查看`labs/lab1/queens.c`源代码：

```
char *usage =
        "Usage:  %s [-ac] n\n\
\tn\tNumber of queens (rows and columns). An integer from 1 to 100.\n\
\t-a\tFind and print all solutions.\n\
\t-c\tCount all solutions, but do not print them.\n";
```

可以通过`-a`参数指定输出问题的所有解、`-c`参数指定输出问题的解的个数，仿真时可以使用`--options`选项将命令行参数传给二进制文件，若想得到其他数量的皇后（如4个）对应的解：

```
$ build/ARM/gem5.debug configs/example/se.py -c labs/lab1/queens --cpu-
type=MinorCPU --caches --l2cache --l1d_size=8kB --l1i_size=8kB --l1d_assoc=4 --
l1i_assoc=4 --l2_size=1MB --l2_assoc=8 --cacheline_size=64  --cpu-clock=1GHz --
num-cpus=1  --options="-a 4"
```

```
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
info: Increasing stack size by one page.
warn:   instruction 'bti' unimplemented
warn: ignoring syscall mprotect(...)
4 queens on a 4x4 board...

Solution #1:
 - Q - -
 - - - Q
 Q - - -
 - - Q -

Solution #2:
 - - Q -
 Q - - -
 - - - Q
 - Q - -

...there are 2 solutions
Exiting @ tick 83652000 because exiting with last active thread context
```

若想得到指定皇后数对应的解的总个数:

```
$ build/ARM/gem5.debug configs/example/se.py -c labs/lab1/queens --cpu-
type=MinorCPU --caches --l2cache --l1d_size=8kB --l1i_size=8kB --l1d_assoc=4 --
l1i_assoc=4 --l2_size=1MB --l2_assoc=8 --cacheline_size=64  --cpu-clock=1GHz --
num-cpus=1  --options="-c 4"
```

```
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
info: Increasing stack size by one page.
warn:   instruction 'bti' unimplemented
warn: ignoring syscall mprotect(...)
4 queens on a 4x4 board...
...there are 2 solutions
Exiting @ tick 75828000 because exiting with last active thread context
```

2.3 交叉编译的目的

为了能够为运行编译器的平台以外的平台创建可执行代码。如在x86平台上使用`aarch64-linux-gnu-gcc`交叉编译器生成可以在arm平台上运行的可执行文件。

2.4 (选做)自行搭建交叉编译器

编译指定架构的处理器:

```
$ scons build/MIPS/gem5.debug -j 4
```

安装ARM指令集交叉编译器`mips-linux-gnu-gcc`:

```
$ sudo apt-get install emdebian-archive-keyring
$ sudo apt-get install linux-libc-dev-mips-cross libc6-mips-cross libc6-dev-mips-cross binutils-mips-linux-gnu gcc-mips-linux-gnu g++-mips-linux-gnu
```

对八皇后程序`queens.c`进行交叉编译：

```
$ mipsel-linux-gnu-gcc -o labs/lab1/queens_mips labs/lab1/queens.c --static
```

> 注意这里必须使用`mipsel-linux-gnu-gcc`而不是`mips-linux-gnu-gcc`，这是由于如果用后者会出现以下错误（gem5仅支持小端法的MIPS）：
>
> ```
> Global frequency set at 1000000000000 ticks per second
> warn: No dot file generated. Please install pydot to generate the dot file and pdf.
> warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)
> 0: system.remote_gdb: listening for remote gdb on port 7000
> **** REAL SIMULATION ****
> info: Entering event queue @ 0.  Starting simulation...
> info: Increasing stack size by one page.
> warn: Prefetching not implemented for MIPS
> warn: 2315000: Halting thread 0 in system.cpu @ PC 422b40, setting restart PC to 422b44
> Exiting @ tick 18446744073709551615 because simulate() limit reached
> ```

根据所提供的参数，在MIPS架构上对可执行程序进行仿真：

```
$ build/MIPS/gem5.debug configs/example/se.py -c labs/lab1/queens_mips --cpu-type=TimingSimpleCPU --caches --l2cache --l1d_size=8kB --l1i_size=8kB --l1d_assoc=4 --l1i_assoc=4 --l2_size=1MB --l2_assoc=8 --cacheline_size=64  --cpu-clock=1GHz --num-cpus=1
```

```
Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and pdf.
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
info: Increasing stack size by one page.
warn: Prefetching not implemented for MIPS
warn: 2315000: Halting thread 0 in system.cpu @ PC 422b40, setting restart PC to 422b44
Exiting @ tick 18446744073709551615 because simulate() limit reached
```

这里并没有运行成功，是由于gem5对于MIPS架构的支持不够全面，并不支持prefench这一操作。

3 流水线处理器的性能分析

3.1 Minor流水线结构

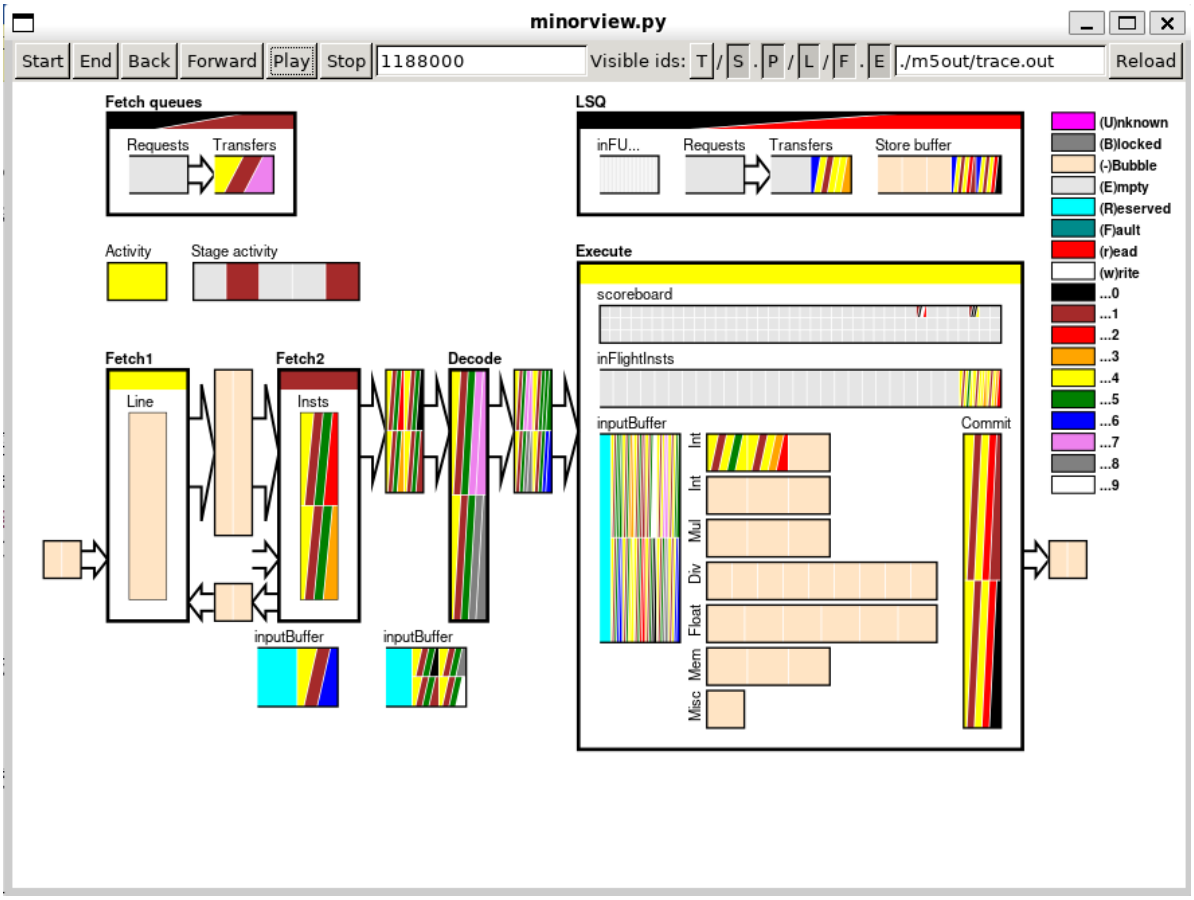
输出trace文件：

```
$ build/ARM/gem5.debug --debug-flags=MinorTrace --debug-start=10000 --debug-file=trace.out configs/example/se.py -c labs/lab1/queens --cpu-type=MinorCPU --caches --l2cache --l1d_size=8kB --l1i_size=8kB --l1d_assoc=4 --l1i_assoc=4 --l2_size=1MB --l2_assoc=8 --cacheline_size=64  --sys-clock=1GHz --num-cpus=1
```

运行可视化：

```
$ python2 ./util/minorview.py ./m5out/trace.out
```

运行截图如下：



a)

可见，Minor处理器由4级流水线构成，每一级的名称和作用如下：

stage number	名称	功能
1	Fench1	instruction fetch unit responsible for fetching cache lines
2	Fench2	line to instruction decomposition
3	Decode	instruction to micro-op decomposition
4	Execute	instruction execution and data memory interface

b)

Execute stage包括以下模块：

- scoreboard: 控制指令的发布。它包含一个flight instructions的数量，这些指令将写入每个通用CPU的整数或浮点寄存器。只有当记分牌上的指令数为0时，指令才会被发出，这些指令将写入一个指令源寄存器。一旦指令被发出，指令的每个目标寄存器的记分牌计数将被递增。
- inFlightInsts: 作为存储in flight instructions的队列，并遵循正确的执行顺序，先进先出。
- inputBuffer: 作为指令的缓冲区，等待提交执行。
- Int,Mul,Div...: 这些模块可能类似于经典MIPS五级流水线中的ALU，执行整型（Intx2、Mul、Div），浮点型（Float）运算，还包括一个Misc单元（Minimal instruction set computer）和一个Mem单元。
- Commit: 提交`inFlightInsts`中的指令。

c)

- forward FIFO: 寄存上一级的数据，显示了当前tick时被推入的数据（在左边），传输中的数据，以及其输出的可用数据（在右边）。
- backward FIFO: （Fetch2和Fetch1之间）显示分支预测的信息，Fench2预测的信息回传给Fench1中的指令在遇到分支时发挥决策。

3.2 处理器频率——性能特性

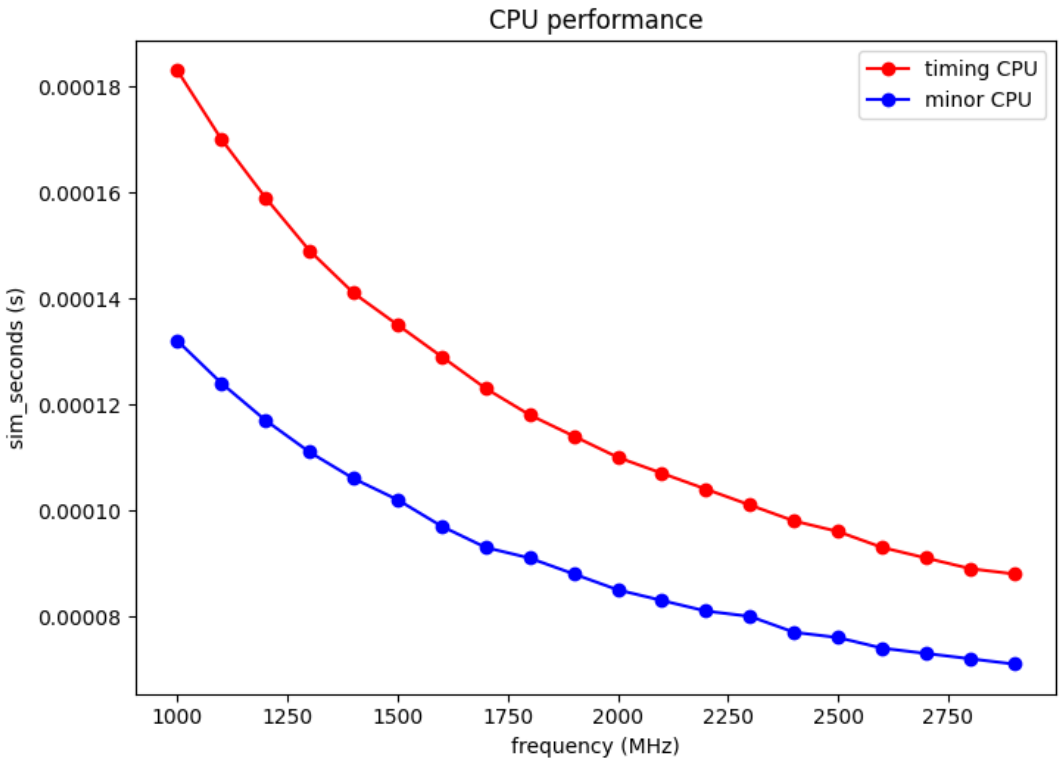
运行脚本如下：

```
# Program:
#     CPU frequency simulation
Target=Frequency_out
Unit=MHz
Root=/home/nullptr/gem5-stable/labs/lab1/"$Target"     # Target dictionary for
stats.txt
Gem5=/home/nullptr/gem5-stable/build/ARM/gem5.debug # gem5 dictionary
Script=/home/nullptr/gem5-stable/configs/example/se.py  # config script
dictionary
Bench=/home/nullptr/gem5-stable/labs/lab1/queens                              #
benchmark name
i=1000 # frequency: MHz
CPUtype=TimingSimpleCPU

mkdir $Root
while [ $i != 3000 ]                    # Max frequency: 3GHz = 3000MHz
do
    # Add your code here for gem5 simulation
    # You can use the following command to run gem5 simulation
    echo $i
    $Gem5 $Script -c $Bench --cpu-type=$CPUtype --caches --l2cache --l1d_size=8kB
--l1i_size=8kB --l2_size=1MB --l1d_assoc=4 --l1i_assoc=4 --l2_assoc=8 --
cacheline_size=64 --sys-clock=${i}${Unit} --num-cpus=1 --
output=$Root/${i}_${CPUtype}
    # add i for 100
    i=$(($i+100))
done;
echo "finished"
exit 0
```

运行仿真：

```
$ ./sim_freq.sh | tee timing.log
$ ./sim_freq.sh | tee minor.log
```

结果如下：



(a)

可以看出，随着频率增长，处理器运行时间变短，且变化速率越来越慢（呈非线性关系）。从1GHz->3GHz，timing CPU提升了约2.08倍，minor CPU提升了约1.86倍。推测原因如下：

CPU的时钟周期中，90%的时间都在访存。CPU的访存瓶颈来源于CPU时钟速度与内存时钟速度无法匹配，换句话说，内存延迟是造成CPU访存瓶颈的罪魁祸首。提升CPU频率之后，由于cache、memory的访存速度没有明显提升，无法与CPU匹配，导致速度被拖慢（设想以下场景：CPU只是频率变高，但执行代码的逻辑并没有发生变化，对于cache、memory的访问逻辑也没有发生改变。这个时候，假如发生了cache未被命中的情况，从memory中提取数据的时间仍然没有缩短，且占据了大量时间，大大抵消了增加主频带来的好处）。

(b)

timing CPU上升更多。这是因为minor CPU是4周期流水线CPU，而timing CPU在每个周期中计算一个指令，只在访存阶段需要等待多个周期读取内存——也就是说，timing CPU在缓存访问时停滞不前，等待内存系统的响应，然后再继续进行。这个时候内存时钟对速度的影响没有那么明显，CPU时钟也成为影响访存速度的一个重要因素。

3.3 处理器频率——性能特性

观察八皇后问题的源程序，发现解决八皇后问题中涉及到的计算均为整数计算，所以推测应该修改整数运算单元。`stats.txt`的统计信息也证实了这一点：

```
system.cpu.op_class_0::No_OpClass                 302      0.67%      0.67% # Class of committed instruction
system.cpu.op_class_0::IntAlu                   28879     64.21%     64.89% # Class of committed instruction
system.cpu.op_class_0::IntMult                      7      0.02%     64.90% # Class of committed instruction
system.cpu.op_class_0::IntDiv                       3      0.01%     64.91% # Class of committed instruction
system.cpu.op_class_0::FloatAdd                     0      0.00%     64.91% # Class of committed instruction
system.cpu.op_class_0::FloatCmp                     0      0.00%     64.91% # Class of committed instruction
system.cpu.op_class_0::FloatCvt                     0      0.00%     64.91% # Class of committed instruction
system.cpu.op_class_0::FloatMult                    0      0.00%     64.91% # Class of committed instruction
system.cpu.op_class_0::FloatMultAcc                 0      0.00%     64.91% # Class of committed instruction
system.cpu.op_class_0::FloatDiv                     0      0.00%     64.91% # Class of committed instruction
system.cpu.op_class_0::FloatMisc                    0      0.00%     64.91% # Class of committed instruction
system.cpu.op_class_0::FloatSqrt                    0      0.00%     64.91% # Class of committed instruction
```

对以下参数进行修改：

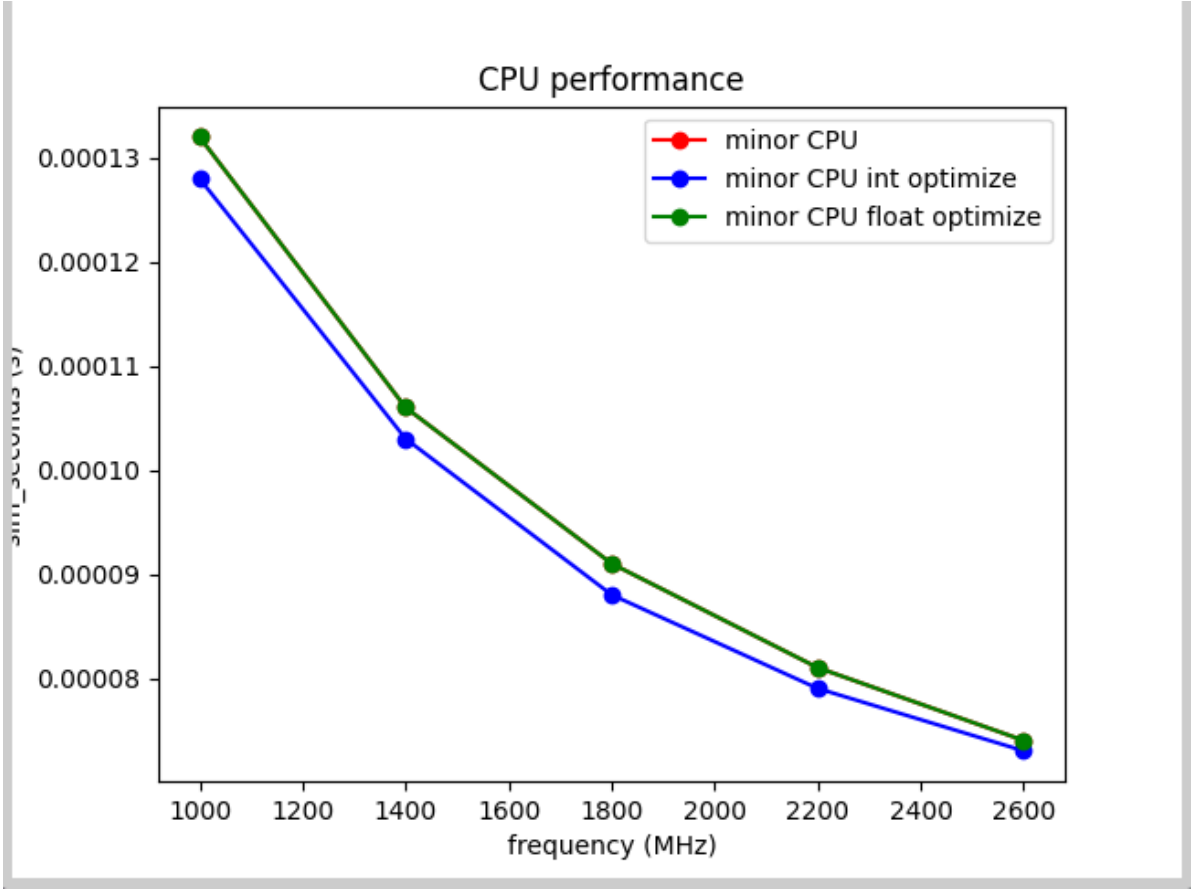
version 1：仅修改整数算子

```
class MinorDefaultIntFU(MinorFU):
    opClasses = minorMakeOpClassSet(['IntAlu'])
    timings = [MinorFUTiming(description="Int",
        srcRegsRelativeLats=[2])]
-    opLat = 3
+    opLat = 1

class MinorDefaultIntMulFU(MinorFU):
    opClasses = minorMakeOpClassSet(['IntMult'])
    timings = [MinorFUTiming(description='Mul',
        srcRegsRelativeLats=[0])]
-    opLat = 3
+    opLat = 1

class MinorDefaultIntDivFU(MinorFU):
    opClasses = minorMakeOpClassSet(['IntDiv'])
    issueLat = 9
-    opLat = 9
+    opLat = 4
```

version 2：仅修改浮点算子

```
class MinorDefaultFloatSimdFU(MinorFU):
    opClasses = minorMakeOpClassSet([
        'FloatAdd', 'FloatCmp', 'FloatCvt', 'FloatMisc', 'FloatMult',
        'FloatMultAcc', 'FloatDiv', 'FloatSqrt',
        'SimdAdd', 'SimdAddAcc', 'SimdAlu', 'SimdCmp', 'SimdCvt',
        'SimdMisc', 'SimdMult', 'SimdMultAcc', 'SimdShift', 'SimdShiftAcc',
        'SimdDiv', 'SimdSqrt', 'SimdFloatAdd', 'SimdFloatAlu', 'SimdFloatCmp',
        'SimdFloatCvt', 'SimdFloatDiv', 'SimdFloatMisc', 'SimdFloatMult',
        'SimdFloatMultAcc', 'SimdFloatSqrt', 'SimdReduceAdd', 'SimdReduceAlu',
        'SimdReduceCmp', 'SimdFloatReduceAdd', 'SimdFloatReduceCmp',
        'SimdAes', 'SimdAesMix',
        'SimdSha1Hash', 'SimdSha1Hash2', 'SimdSha256Hash',
        'SimdSha256Hash2', 'SimdShaSigma2', 'SimdShaSigma3'])

    timings = [MinorFUTiming(description='FloatSimd',
        srcRegsRelativeLats=[2])]
-    opLat = 6
+    opLat = 3
```

优化后的结果如下：



可以看出，只有修改整数运算单元时，才能提升计算速度。这与之前的推测相一致。

参考文献