lab 4实验报告

20302010043 苏佳迪

一、流水线的改动

为了支持异常处理，流水线做出以下调整：

1、csr指令支持

流水线需要支持 `csrrw` 等指令执行写入后刷新，刷新即清除当前流水线中的所有指令（流水寄存器 `reset` ），同时设置从 `csr` 指令的下一条开始执行（pc跳转）。具体实现为：

实现**刷新**：当 `dataW.csrwrite` 为高位时，说明当前指令需要写入 `csr` ，此时该周期需要把 `fetch_decode` （清除下个周期译码段）， `decode_execute` （清除下个周期执行段） `execute_memory` （清除下个周期访存段）与 `memory_writeback` （清除下个周期写回段）寄存器的 `reset` 信号拉高，完成下个周期的流水线刷新。同时**当W阶段的指令为 `csr` 指令时，避免 `memory` 的访存操作，避免刷新前写入错误的数据。**

实现**pc的更新**：当 `dataW.csrwrite` 为高位时，预期为下个周期流水线被刷新，同时 `fetch` 阶段开始 `dataW.pc + 4` 的取指，也就是在当前周期就应该将 `dataW.pc + 4` 设置为 `pcselect` ，进行下个周期的pc更新，且这个选择的优先级要高于 `dataE.addr` 的跳转地址的，因为一个周期中， `pcselect` 接收到的非顺序pc只有两个来源 `execute` 与 `writeback` ，前者只是普通的跳转，优先级最低。因此当同时出现 `execute` 与 `writeback` 的跳转pc请求，响应后者的。

同时有一种情况：当前周期 `dataW.csrwrite` 为高电位，说明下周期重新开始执行指令，但如果该周期 `pc` 还没有握手，下周期的pc不会更新，而 `writeback` 会流出流水线，就导致在fetch取指完成后无法成功跳到 `dataW.pc + 4` 的位置，所以如果要刷新流水线时的fetch还在取指令，就阻塞整个流水线（包括 `writeback` ）不让其流动，等到 `iresp.data_ok` 为1时不再阻塞，某个周期 `data_ok` 时，将阻塞信号置为0，下个周期 `writeback` 流出，同时 `fetch` 的pc更新为 `dataW.pc + 4` 完成流水线的刷新。

2、进入异常

异常会在每个流水段检测，当 `dataW.exception` 为高位时，说明该指令出现异常需要处理，此时该周期需要把 `fetch_decode` （清除下个周期译码段）， `decode_execute` （清除下个周期执行段） `execute_memory` （清除下个周期访存段）与 `memory_writeback` （清除下个周期写回段）寄存器的 `reset` 信号拉高，完成下个周期的流水线刷新。同时**当W阶段的发生异常时，避免 `memory` 的访存操作，避免刷新前写入错误的数据。**

实现**pc的更新**：当 `dataW.exception` 为高位时，预期为下个周期流水线被刷新，同时 `fetch` 阶段开始 `mtvec` 的取指，也就是在当前周期就应该将 `mtvec` 设置为 `pcselect` ，进行下个周期的pc更新，且这个选择的优先级要高于 `dataE.addr` 的跳转地址与 `csr + 4` 的刷新地址，此时一个周期中， `pcselect` 接收到的非顺序pc只有两个来源 `execute` 与 `writeback` ，前者只是普通的跳转，优先级最低，后者可能出现两个情况： `csr` 指令导致的刷新（ `dataW.pc + 4` ）与发生异常导致的刷新（ `mtvec` ），后者优先级高。**但如果该周期 `pc` 还没有握手**，下周期的pc不会更新，而 `writeback` 会流出流水线，就导致在fetch取指完成后无法成功跳到 `mtvec` 的位置，所以如果要刷新流水线时的fetch还在取指令，就阻塞整个流水线（包括 `writeback` ）不让其流动，等到 `iresp.data_ok` 为1时不再阻塞，某个周期 `data_ok` 时，将阻塞信号置为0，下个周期 `writeback` 流出，同时 `fetch` 的pc更新为 `mtvec` 完成流水线的刷新。

3、离开异常

逻辑与进入异常相同，检测从 `exception` 改为 `mret` 。

4、响应中断

首先是中断与处理中断信号的检测：中断信号是由外部随机时刻到来的，到来后持续到该中断被处理。因此需要考虑中断处理信号的设置：中断处理信号即标志开始处理中断的信号，当某个周期该信号为高电位时，下个周期将进入中断处理。

中断处理信号：根据外部的中断信号与内部的流水线状态决定是否需要开始中断。当某个中断信号为1时，说明在流水线允许的状态下下个周期开始中断，需要刷新流水线。不能进入中断的情况有以下几种：

- 处理器全局中断使能未开启（ `csr.mstatus.mie == 0` ）
- 流水线不便于无法刷新（包括 `ireq` 和 `dreq` 未握手的情况，需要等待流水线访存的握手后再处理）
- 流水线W阶段的下条有效指令pc不便于确认，因为若下周期处理中断的话，当前W阶段的指令将是最后提交的一条指令，当中断处理结束后需要返回到下一条指令继续执行。有以下几种情况：（1） `dataW.instruction` 无效，需要等待有效的下个周期再处理中断；（2）流水线中的指令为错误指令，即存在跳转的情况，包括 `jal` 等指令的执行阶段跳转与更新csr导致的写回阶段跳转，则等待跳转完成后再处理中断，（此时也可以更新pc但需要需要设置 `mepc` 为跳转的目标地址，但似乎不太方便，若有异常除外）。

二、测试通过截图

问题 输出 JUPYTER 调试控制台 终端 wsl

```
Single test passed.
Run paint
Aptenodytes patagonicus
Picture generated.
Compressed size=2154
Done! Result:
```


```
data:image/bmp;base64,Qk1qCAAAAAAAAHYAAAAoAAAAZAAAAGQAAAABAAQAAgAAAPQHAACIEwAAiBMAABAAAAAQAAAAKp55AP///wAhIxAAAK/3ACgNEQB75f8AAAAAAB0nGwA+LygARTUuAFp
PPgBiVkQAppeDAMu8qQDO4c4A9ebTAAYAAmsQuwjMAmIEIgJhIxECYBcAAAAGAAJrErsGzAJiBCICYSIRAmYYAAAABgACaxO7BcwCYgQiAmEiEQJgGAAAAAYAAmoHqg67A8wCZgQiAmEiEQJgGAAA
AAYAAmoLqgu7A8wCYgMiAmEhEQJgGQAAAAYAAmoNqgq7AswCYgMiAmEhEQJgGQAAAAYAAmoPqgm7AsYFIgJhHxECYBoAAAAHAAJqD6oJuwLGBCICYR8RAmAaAAAABwACahGqCLsCYgMiAmEeEQJmG
wAAAAcAAmkGmQyqB7sCYgMiAmEeEQJgGwAAAAcAAmkJmQqqB7sCYgMiAmEdEQJgGwAAAAcAAmkLmQmqBrsCYgMiAmEcEQJgHAAAAAcAAmkMmQiqBrsCYgMiAmYcEQJgHAAAAAcAAmkNmQiqBrsCYg
MiAmEaEQJmHQAAAAcAAmkOmQiqBrsCYgIiAmYaEQJgHQAAAAcAAmgGiAmZB6oGuwJiAyICYRgRAmAeAAAABwACaAiICJkHqga7AmIDIgJhFxECYB4AAAAHAAJoCYgImQaqBrsCxgMiAmEWEQJmHwA
AAAcAAmgKiAeZB6oGuwJiAyICYRURAmAfAAAABwACaAuIB5kGqga7AsYEIgJhFBECYB8AAAAIAAJoC4gGmQaqBrsCzAJiAiICZhMRAmAgAAAACAACZwR3B4gGmQaqBrsDzAJiAiICZhIRAmAgAAAA
CAACZwV3B4gGmQaqBbsEzAJiAiICZhARAmYhAAAACAACZwZ3BogGmQaqBrsEzAJiAiICZg8RAmAhAAAACAACZwZ3BogGmQaqBrsFzAJiAyICZg0RAmAhAAAABwACZgd3BogGmQaqBrsFzALWAmICI
gJmCxECYCIAAAAHAAJmB3cGiAaZBqoGuwXMA90CZgIiAmYKEQJgIgAAAAgAAmcGdwaIBpkGqga7BcwE3QNmAgADZgYRAmAjAAAACAACZwZ3BogGmQaqBrsFzAXdAuYCZgIAA2YEEQJgIwAAAAgAAm
cGdwaIBpkGqga7BcwF3QPuA2YDAAVmJAAAAAgAAmYFdweIBpkGqgW7BswF3QXuAvYGZiUAAAAJAAJnA3cHiAaZBqoGuwbMBd0F7gT/BGYlAAAACQACaAqIBpkGqga7BswF3QXuA/8CZgJEAmAkAAA
ACQACaAmIB5kGqga7BcwG3QXuAvYCZANEAmAkAAAACgACaAeIB5kHqga7BcwG3QTuAmYGRAJgJAAAAAoAAmgGiAiZBqoGuwbMBd0E7gJmB0QCYCQAAAAKAAJmBYgImQeqBrsGzAXdAuYCZglEAmAk
AAAACwACaAKICZkHqge7BcwG3QJkC0QCYCQAAAALAAJpCpkIqga7BswE3QJmDUQCYCQAAAAMAAJpCJkIqge7BswC3QJmAjYNRAJgJAAAAAwAAmkHmQiqB7sGzALdAmMCMwJkDEQCYCQAAAAMAAJmB
pkJqge7BswCZgQzAmQMRAJgJAAAAA0AAmkDmQqqB7sEzANmBjMCZAxEAmAkAAAADQACZgyqB7sEzAJjCDMCZAxEAmAkAAAADgACagqqCLsCzAJmCjMCZAxEAmAkAAAADgACZgiqCbsCxgNmCjMCZg
tEAmAlAAAADwACagaqCbsCxgJEAmMLMwJkCkQCYCUAAAAQAAJqA6oJuwNmAkQCZgwzAmQKRAJgJQAAABAAAmYCqwi7AmYERAJmDTMCZApEAmAlAAAAEQACawi7AmYFRAJjDTMCZApEAmAlAAAAEgA
CawW7AmYGRAJmDjMCZApEAmAlAAAAEgACZgO7AmYIRAJjDjMCZApEAmAlAAAAEwACawK2AmQIRAJjDjMCZApEAmAlAAAAEwADZgpEAmYOMwJmC0QCYCUAAAAUAAJkCkQCYw4zAmQLRAJgJQAAABQA
AmYKRANmDDMCZgxEAmAlAAAAFQACZAxEBWYFMwNmDUQCYCUAAAAWAAJkEEQGZg9EAmAlAAAAFwACZCREAmAlAAAAGAACZCNEAmAlAAAAGAACZiJEAmYmAAAAGQACZiFEA2YlAAAAGgACZiBEA2UlA
AAAHAACZB5EA2UCYCMAAAAdAAJmHUQCZQJWAmAiAAAAHgACZhxEAmUCVQJgIgAAACAAAmYbRAJlAlYCYCEAAAAhAANmGUQCZQJVAmAhAAAAIwADZhdEAmUCVQJmIQAAACUABGYURAJmA1UCYCAAAA
AoAAVmEUQCZQJVAmYgAAAALQAHZgtEAmUCVQJgHwAAADMAAmYKRAJmAlUCZh8AAAA0AAJmCkQCZQJVAmYeAAAANQADZghEAmYDVQJgHQAAADcAAmYIRAJmAlUCZh0AAAA4AANmB0QCZgJVAmAcAAA
AOgACZghEA2UCYBsAAAA7AAJmCEQDZhwAAAA8AANmB0QDZhsAAAA+AAJmCEQCZhoAAAA/AAJmCEQCYBkAAABAAANmBkQCZhkAAABCAAJmBkQCYBgAAABDAANmBEQCZhgAAABFAAJmBEQCYBcAAABG
AAJmA0QCZhcAAABHAANmAkQCYBYAAABJAAJmAkYCYBUAAABKAAVmFQAAAEwAA2YVAAAABwACYAIAA2ACZgIGAmY5AANmFAAAAAcAAmACAAJgAgACYAIAAmA5AAJmFAAAAAcAA2ACZgIGAmYCBgJmU
AAAAAcACmACAAJgTwAAAAcAB2ACZgIGAmZQAAAAZAAAAGQAAABkAAAAZAAAAAAB
```


```
Run coremark
Running CoreMark for 10 iterations
2K performance run parameters for coremark.
CoreMark Size    : 666
Total time (ms)  : 363
Iterations       : 10
Compiler version : GCC11.1.0
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0xfcaf
Finised in 363 ms.
==================================================
CoreMark Iterations/Sec 27
Run dhrystone
Dhrystone Benchmark, Version C, Version 2.2
Trying 10000 runs through Dhrystone.
Finished in 855 ms
==================================================
Dhrystone PASS         20 Marks
                   vs. 100000 Marks (i7-7700K @ 4.20GHz)
Run stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
```


问题 输出 JUPYTER 调试控制台 终端 wsl

```
Dhrystone PASS         20 Marks
                   vs. 100000 Marks (i7-7700K @ 4.20GHz)
Run stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
```


```
Array size = 2048 (elements), Offset = 0 (elements)
Memory per array = 0.0 MiB (= 0.0 GiB).
Total memory required = 0.0 MiB (= 0.0 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
* checktick: start=0.928890
* checktick: start=0.930134
* checktick: start=0.931153
* checktick: start=0.932237
* checktick: start=0.933241
* checktick: start=0.934370
* checktick: start=0.935544
* checktick: start=0.936752
* checktick: start=0.937831
* checktick: start=0.938957
* checktick: start=0.940070
* checktick: start=0.941152
* checktick: start=0.942208
* checktick: start=0.943390
* checktick: start=0.944490
* checktick: start=0.945489
* checktick: start=0.946679
* checktick: start=0.947829
* checktick: start=0.948961
* checktick: start=0.950084
Your clock granularity/precision appears to be 68 microseconds.
Each test below will take on the order of 7594 microseconds.
   (= 111 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:              12.5     0.002822     0.002620     0.002901
Scale:              0.8     0.041796     0.041478     0.042006
Add:                1.1     0.046697     0.045742     0.047463
Triad:              0.5     0.107476     0.106913     0.108200
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
Run conwaygame
```

问题 输出 JUPYTER 调试控制台 终端 wsl

```
Run conwaygame
Play Conway's life game for 200 rounds.
seed=3394
                **
                **     **            ***
           *           **
           *
           *
                            *
                           * *
                          *  *
                           **      ** **
                                   ** **
Run sys-test
trap here, epc 8000600c, cause 8
Test ecall_u [OK]
trap here, epc 8000608c, cause 8
trap here, epc 8000602e, cause 0
Test instr_misalign [OK]
trap here, epc 8000608c, cause 8
trap here, epc 80006040, cause 4
Test load_misalign [OK]
trap here, epc 8000608c, cause 8
trap here, epc 80006050, cause 6
Test store_misalign [OK]
trap here, epc 8000608c, cause 8
trap here, epc 800060bc, cause 8000000000000007
Test timer_intr [OK]
trap here, epc 8000608c, cause 8
trap here, epc 80006090, cause 8000000000000003
Test software_intr [OK]
trap here, epc 8000607c, cause 8
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
```

问题 输出 JUPYTER 调试控制台 终端 wsl

```
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Timer interrupt in test_trap, this should happen 50 times.
Test m_trap [OK]
Privileged test finished.
Exit with code = 0
^CCore 0: SOME SIGNAL STOPS THE PROGRAM at pc = 0x0
total guest instructions = 0
instrCnt = 0, cycleCnt = 0, IPC = -nan
Seed=0 Guest cycle spent: 365998807 (this will be different from cycleCnt if emu loads a snapshot)
Host time spent: 331560ms
This emulator compiled with JTAG Remote Bitbang client. To enable, use +jtag_rbb_enable=1.
Listening on port 23334
make: *** [Makefile:60: test-lab4] Interrupt
```