AIAS Spring 2023 \N26112437_\劉兆軒_期末上機考Bonus

上機考Bonus 的執行時間限定在 6/16 9:10am-12:00pm, 超過時間之後，你還是可以做, 但是就不算在上機考Bonus 裡, 請大家在4:00pm 之後就不要再更新你的作答文件, 我們採榮譽制度，請大家遵守考試規則, 這裡的Bonus 都是為了測試同學的作業答案的robustness, 理論上大家都寫過作業, 所以你應該都很熟悉自己的程式, 如果老師給的新的測資讓你的實作fail, 那你要debug, 修正你的實作, 你可以copy & paste 這份文件成作答文件, 並記得修改你的學號姓名, 如果你有任何test fail 要修正, 請在Gitlab 你的repo 上, 不要更動你原先的設計, 建立一個 final_bonus 的branch, 把你的修正做在final_bonus branch 上

```
$ cd <your lab repo>
$ git checkout -b final_bonus
```

完成Bonus 的要求如下：

- 把code 推到你的gitlab repo
- 把code link 放在你的codimd 文件
- 把對應Bonus 的藍色info 部分要填上答案

請在6/16 12:00pm 把你的文件link 繳交到作業中心- 期末上機考Bonus
雖然我們的作業中心過了12:00pm 還是可以收文件, 但是助教老師評分時, 會以繳交時間為限, 另外是過了12:00pm 之後的所有文件的更新都不算數

作答結束之後請繳交至作業中心繳交link (https://playlab.computing.ncku.edu.tw:8000/course/homework?id=154)

Bonus #1 HW 2-4 AI model Statistics

你的作業是分析lenet.onnx 裡的conv2D operators, 請下載並分析GoogleNet-Int8 model (https://github.com/onnx/models/blob/main/vision/classification/inception_and_googlenet/googlenet/model/googlenet-12-int8.onnx), calculate the number of multiplication operations in each Conv operator and print it out.

Gitlab code Link

Please paste your code link here.
記得註記你的commit number

Execution Result

Please compile and run the program.
Take a screenshot of the execution result and paste it here.

Bonus #2 HW3-3 2x2 Sudoku

請確定下面的測資能被正確解出

```
# You can modify your own test data here.
char test_c_data[16] = { 0, 3, 0, 0,
                         0, 0, 1, 0,
                         1, 0, 0, 2,
                         0, 1, 0, 0 };

char test_asm_data[16] = { 0, 3, 0, 0,
                         0, 0, 1, 0,
                         1, 0, 0, 2,
                         0, 1, 0, 0 };
```

Gitlab code Link

Please paste your code link here.
記得註記你的commit number

Simulation Result

請放上你在 docker中的 rv32emu模擬結果，驗證程式碼的正確性。 (螢幕截圖即可)

Bonus #3 HW4-1 - RISC-V M-Standard Extension

請測試下面的assembly code是否在你的design 上能正確運行

```
main:
lui     x28, 255
addi    x28, x28, 2047
addi    x28, x28, 2047
lui     x29, 255
addi    x29, x29, 2047
addi    x29, x29, 2047
mulhu   x30, x28, x29
mul     x29, x28, x29
mulhsu  x30, x30, x29
lui     x29, 255
addi    x29, x29, 2047
addi    x29, x29, 2047
mulhu   x30, x28, x29
mulhsu  x29, x28, x29
rem     x30, x28, x29
remu    x29, x28, x30
lui     x28, 255
addi    x28, x28, 2047
addi    x28, x28, 2047
mulhu   x30, x28, x29
mulhsu  x29, x28, x29
mulhsu  x30, x30, x29
rem     x30, x28, x29
remu    x29, x28, x30
hcf
```

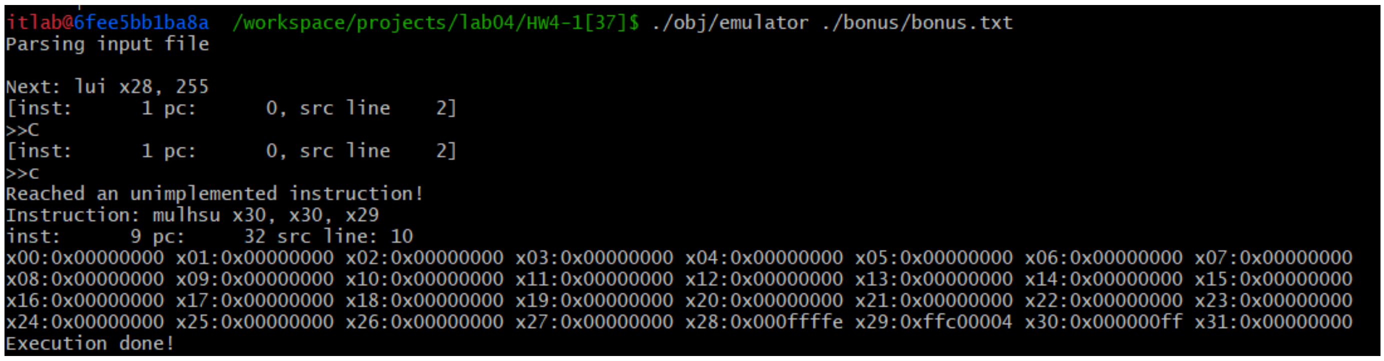
Gitlab code Link

> Please paste your code link here.
> 記得註記你的commit number

https://playlab.computing.ncku.edu.tw:4001/kevin1217/lab4/tree/final_bonus
(https://playlab.computing.ncku.edu.tw:4001/kevin1217/lab4/tree/final_bonus)

Simulation Result

> 請放上
> 1. 你在 docker中的 rv32emu模擬結果，驗證程式碼的正確性。 (螢幕截圖即可)
> 2. 跟venus emulator 執行的結果比較, 一致正確嗎？



```
itlab@6fee5bb1ba8a  /workspace/projects/lab04/HW4-1[37]$ ./obj/emulator ./bonus/bonus.txt
Parsing input file

Next: lui x28, 255
[inst:      1 pc:      0, src line    2]
>>C
[inst:      1 pc:      0, src line    2]
>>c
Reached an unimplemented instruction!
Instruction: mulhsu x30, x30, x29
inst:      9 pc:     32 src line: 10
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0x000ffffe x29:0xffc00004 x30:0x000000ff x31:0x00000000
Execution done!
```

Test Result

> 請放上你通過test的結果，驗證程式碼的正確性。(螢幕截圖即可)

Bonus #4 Hw6-2-3 Order of Operation (+、-、*、(、))

請測試下面的三個測資看看能否在你的設計上正確運行

```
Input1：((-123) * ((-32)+3)*4 + ( 15-(-16)))*(((-4) -2) * ((-2) + 1))
Input2: (5+3)*(7-8) + ((-3)*(7+3) *(8+9) - 7)*(( 3+ 4) - 3)
Input3: ((((((((8-3)*2-4)*3-2)*4-1)*3+5)*2-1)*3+2)*2+4)*3+8)*4 - 1234567890*98271811
```

Notes

Tips

1. Remeber the assumptions in the design, so when you want to reuse the design, you can know how to fix it.
2. If the proplem is much more complex, the state graph should be more states to handle the complex situations.
3. For processing the stream, we can design the state graph like this to force the input stream match the rule we made, so we can make the situation in our control.

Test Result

請放上你通過test的結果，驗證程式碼的正確性。(螢幕截圖即可)

Bonus #5 HW7-1 & HW7-2

下面是一個binary search 的程式,請在你的single-cycle CPU 測試看看能否跑完, 並且達到下面的結果:

- return index at $a5,if search element not found , put -1 in a5

```
# s0: base address of array
# s1: array size
# s2: the search element
# return index at $a5,if search element not found , put -1 in a5

.data
success1: .string "found  "
success2: .string " at index "
fail_string1: .string "search element "
fail_string2: .string " not found "
array:   .word 1, 3, 4, 6, 9, 12, 14, 15, 17, 19, 24
array_size: .word 11
search_element:.word 13
.text
main:
    # binarySearch(int arr[], int l, int r, int x)
    addi t0, zero, 0    #int l
    la   t1, array_size
    lw   t1, 0(t1)      #int r
    addi t1, t1, -1
    add  s10,t1,zero   # s10 used to check if search is go out of bound of array
    addi a5, zero, -1
    la   s2, search_element
    lw   s2, 0(s2)              #s2 stands for search value
    la   s0, array           # load the base address to $s0
    jal  ra, binary_search      # Jump-and-link to the 'binarySearch' label
    bltz a5, Fail                             #check if found or not
    j exit
binary_search:
    # a2 stand for mid = (l + r)/2
    add  a2, t0, t1
    srli a2, a2, 1
    #check if mid > array_size
    blt s10,a2,Fail
    #check if mid < 0
    bltz a2,Fail
    # check if array[mid]== search_element
    add  t2, a2, zero
    slli t2, t2, 2 # t2=t2*4
    add  t2, t2, s0
    lw   t2, 0(t2)
    beq  t2, s2, Find
    # check if to == t1 and still not equal
    beq t0,t1,Fail
    #not equal then adjust l,r depends on the value of array[mid]
    blt  s2, t2, less
greater: #elseif target > array[mid] : l = mid + 1
    addi t0,a2,1
    j binary_search
less: # @if target<array[mid] : r = mid-1
    addi t1,a2,-1
    j binary_search
    ret
Fail:
    addi a5,zero,-1
    la  a1, fail_string1
    li  a0, 4
    ecall

    mv       a1, s2
    li       a0, 1
    ecall

    la  a1, fail_string2
    li  a0, 4
    ecall
    j exit
Find:
    add a5,a2,zero
    la  a1, success1
    li  a0, 4
    ecall

    mv       a1, s2
    li       a0, 1
    ecall

    la  a1, success2
    li  a0, 4
    ecall

    mv       a1, a5
    li       a0, 1
    ecall

    j exit
exit:
```

Gitlab code Link

Please paste your code link here.
記得註記你的commit number

Simulation Result

請放上

1. 你在 docker中的 rv32emu模擬結果，驗證程式碼的正確性。 (螢幕截圖即可)
2. 跑Single-Cycle CPU chisel模擬之後的結果