UCCS University of Colorado Colorado Springs

CS 4200, Spring 2022: Computer Architecture

Instructor: Mong Sim

Project Three

Sean Campbell

2022/04/23

Project Three: Debugger

Discussion:

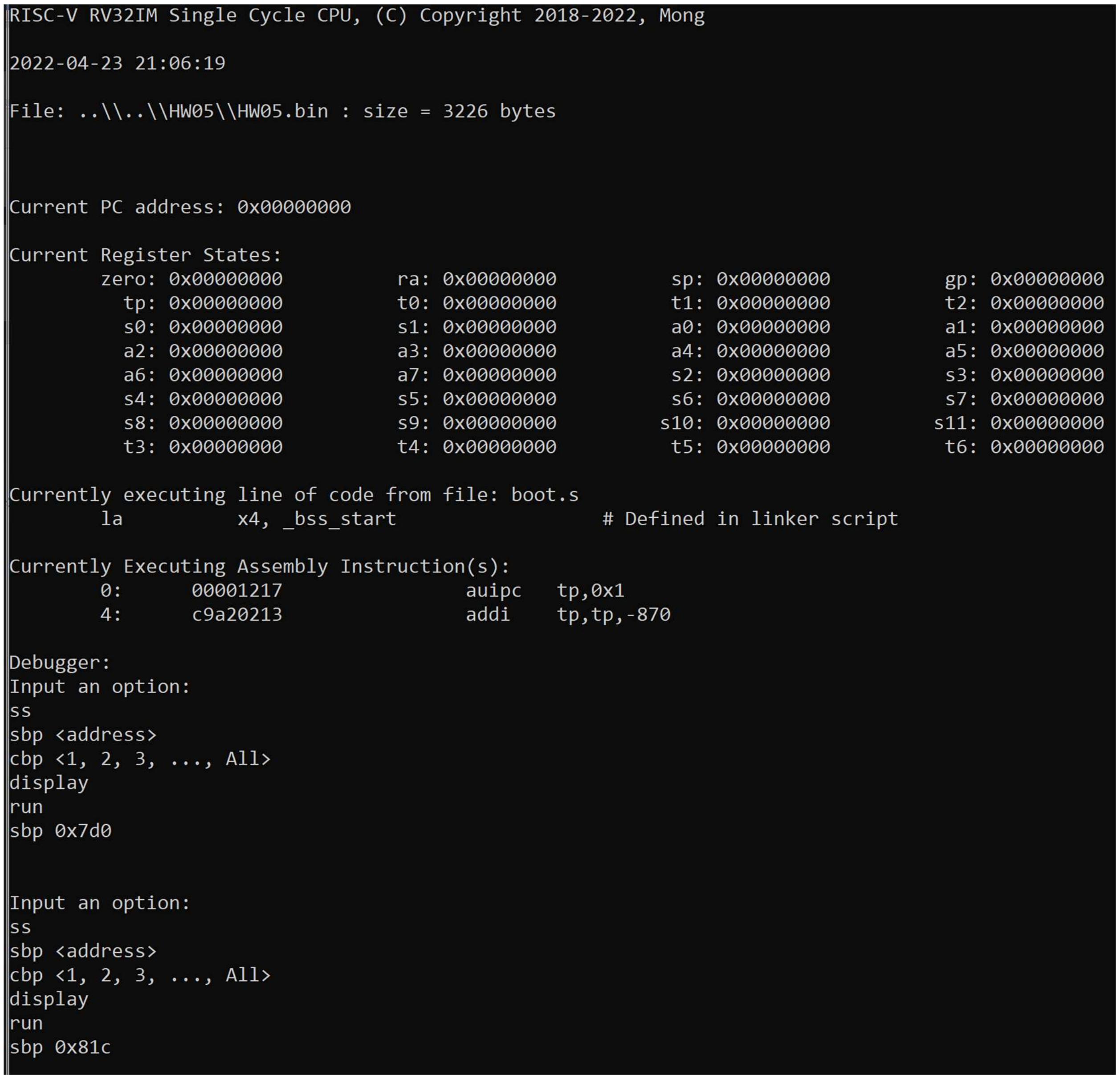
From a high level, this is a project that seems like it would be very difficult to complete, but while working on it, I ended up only running into a couple of problems. My debugger is very straight forward. It is called after the IF_Stage() only if it is the start of the program, the keyboard is hit, or if the current instruction is an EBREAK (0x100073). The first thing that my debugger does is print the current program counter address, the current register states, the currently executing line of C code (or boot.s code if it is the initial break), and finally the line(s) of assembly code that make up the C code. Then, the user is prompted with five different input options. They can either single step to the next line of C code, set a break point at a given address, clear a specific breakpoint, clear all of the break points, display all of the set breakpoints, or finally, run the program until the next breakpoint is hit. The current state of the program is only output when either single stepping or running, the user is allowed to clear or set an unlimited number of breakpoints without having the program step to the next line of code between every input.

The first instruction I implemented was the single step instruction. This instruction is very simple, all that happens is the current instruction is passed back to the core, as long as it is not an EBREAK, and the start flag is set back to a value of 1 so that the debugger will automatically be called on the next iteration. The run instruction works exactly opposite of that, it passes the current instruction back to the core, but now that start flag is set to 0 so that the debugger will not automatically be called again. The display instruction is a simple process that just prints the contents of the breakpoint table, or will display a message stating that there are no breakpoints if none are currently set. The set breakpoint instruction will first, load the instruction into the breakpoint table. Then, it loads the main memory with an EBREAK instruction at the associated memory address. The most difficult instruction to implement was the clear break point instruction. For this instruction, I first remove the breakpoint that needs to be cleared. Then, I update the main memory so that it contains the original instruction. Lastly, I need to worry about the instruction that was passed to the debugger. If this instruction was an EBREAK of a breakpoint that is being cleared, I need to pass the original instruction back to the core, if this was not the case, then I can proceed as normally.

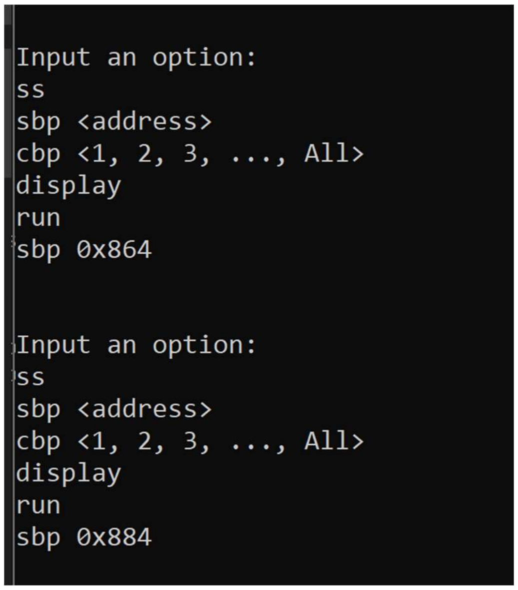
One of the main problems that I ran into was how to access the main memory and the disassembler function from the core class. These are both needed in the debugger, but can only be accessed through an instance of that class. To solve this problem, I passed an object of that class to my debugger function. To make sure that I have the CPU0 object that was created in main, I pass the "this" keyword to my function. The “this” keyword is a keyword that represents the current instance of the class. The second problem that I ran into was how to pass the correct instruction back to the core, while leaving the EBREAK in main memory. To solve this problem, I decided to pass the instruction variable to my debugger as a pointer. This is so that I can always rewrite the instruction with the real instruction if I need to, but leave the EBREAK in main memory.

OUTPUT BELOW:

Setting 4 breakpoints

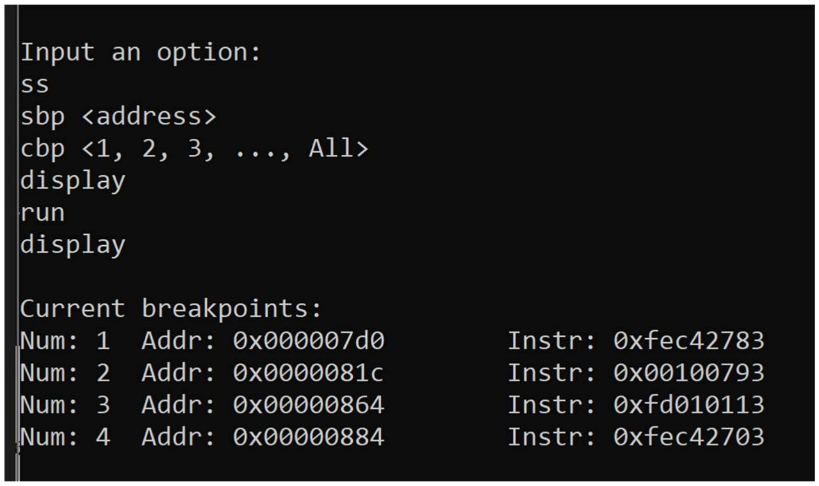


```
RISC-V RV32IM Single Cycle CPU, (C) Copyright 2018-2022, Mong

2022-04-23 21:06:19

File: ..\\..\\HW05\\HW05.bin : size = 3226 bytes



Current PC address: 0x00000000

Current Register States:
        zero: 0x00000000          ra: 0x00000000          sp: 0x00000000          gp: 0x00000000
          tp: 0x00000000          t0: 0x00000000          t1: 0x00000000          t2: 0x00000000
          s0: 0x00000000          s1: 0x00000000          a0: 0x00000000          a1: 0x00000000
          a2: 0x00000000          a3: 0x00000000          a4: 0x00000000          a5: 0x00000000
          a6: 0x00000000          a7: 0x00000000          s2: 0x00000000          s3: 0x00000000
          s4: 0x00000000          s5: 0x00000000          s6: 0x00000000          s7: 0x00000000
          s8: 0x00000000          s9: 0x00000000         s10: 0x00000000         s11: 0x00000000
          t3: 0x00000000          t4: 0x00000000          t5: 0x00000000          t6: 0x00000000

Currently executing line of code from file: boot.s
        la          x4, _bss_start                  # Defined in linker script

Currently Executing Assembly Instruction(s):
        0:      00001217                auipc   tp,0x1
        4:      c9a20213                addi    tp,tp,-870

Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
sbp 0x7d0


Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
sbp 0x81c
```



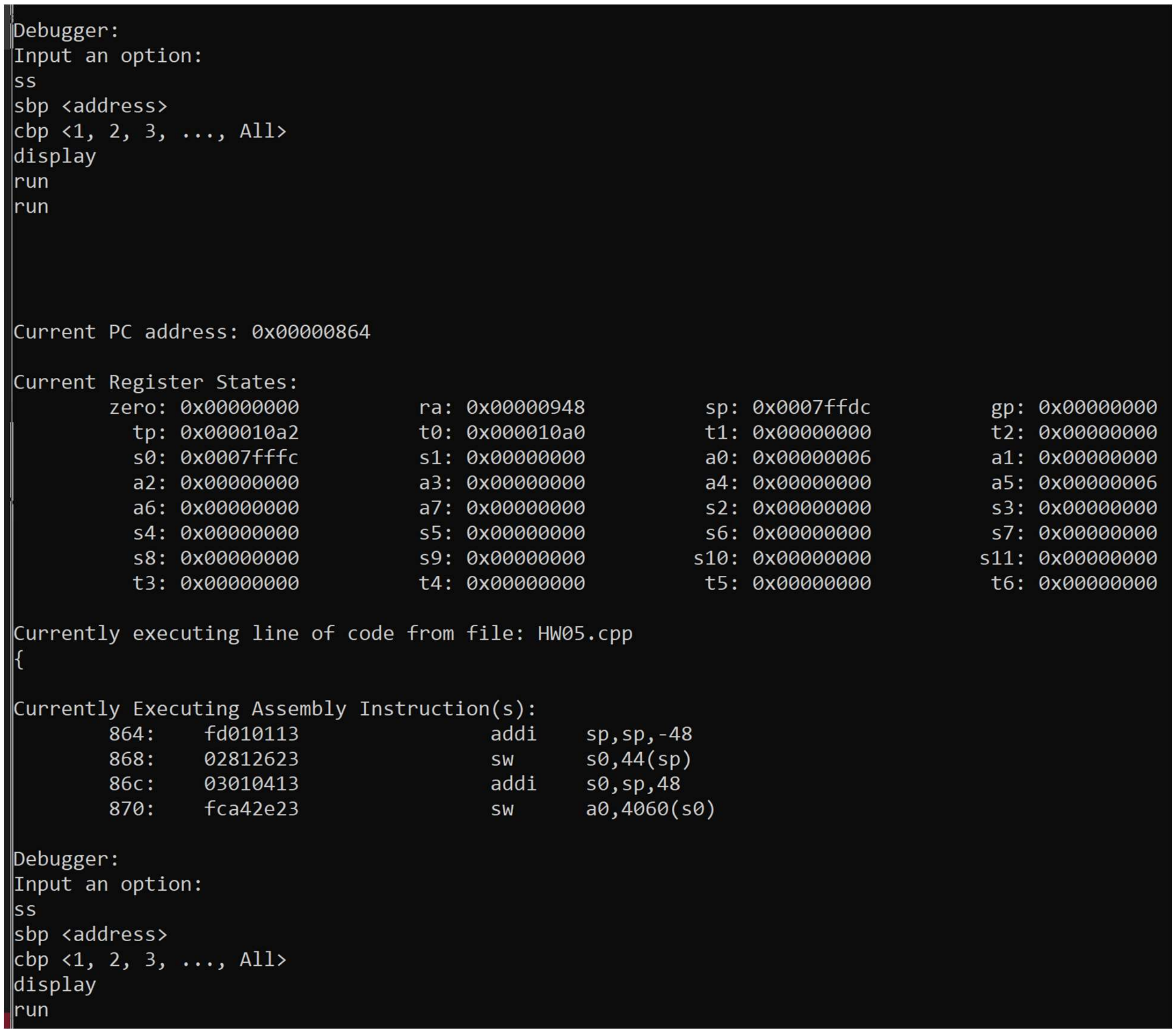
```
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
sbp 0x864


Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
sbp 0x884
```

Displaying All Set Breakpoints



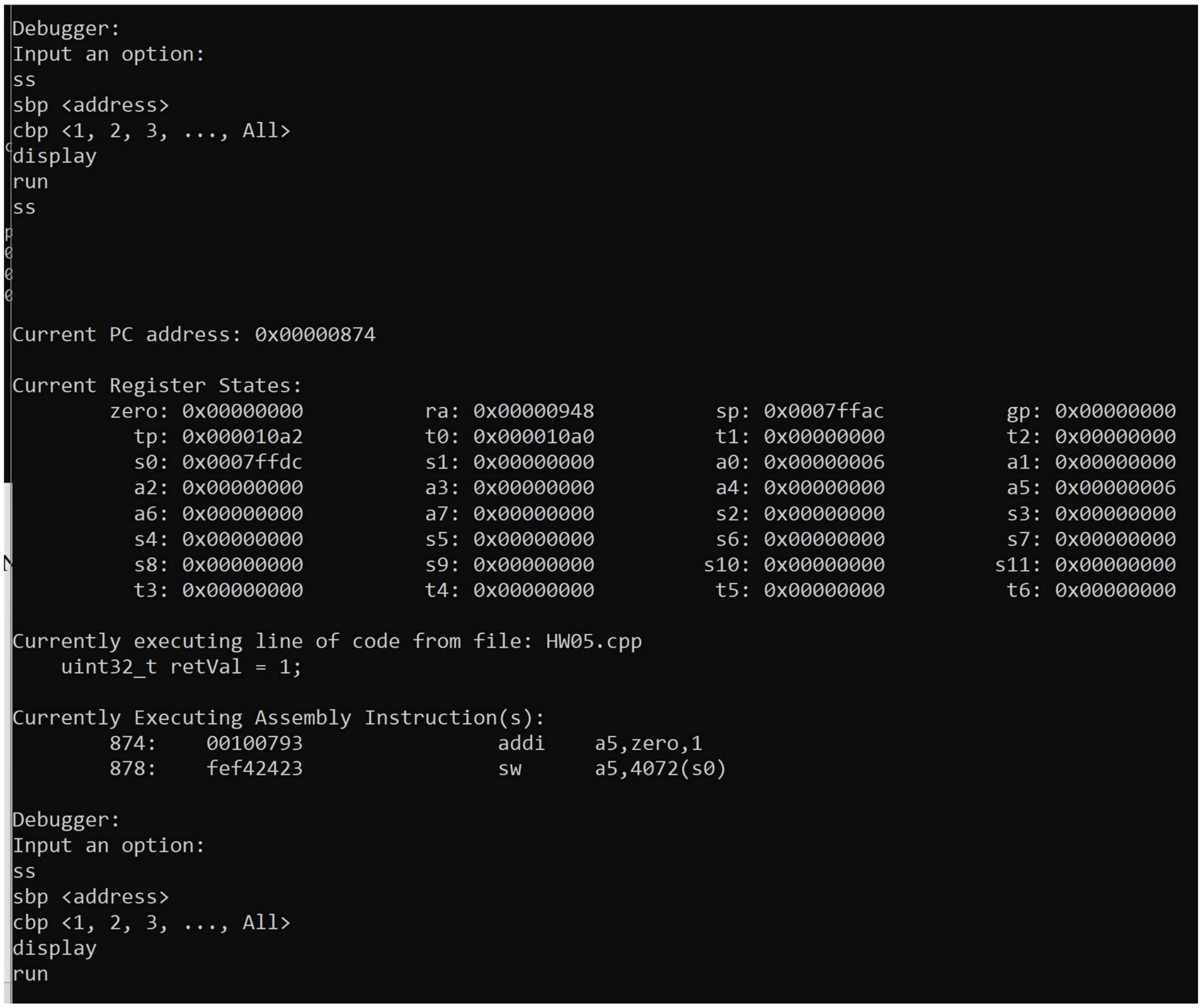
```
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
display

Current breakpoints:
Num: 1  Addr: 0x000007d0        Instr: 0xfec42783
Num: 2  Addr: 0x0000081c        Instr: 0x00100793
Num: 3  Addr: 0x00000864        Instr: 0xfd010113
Num: 4  Addr: 0x00000884        Instr: 0xfec42703
```

Running to Until Breakpoint is Hit



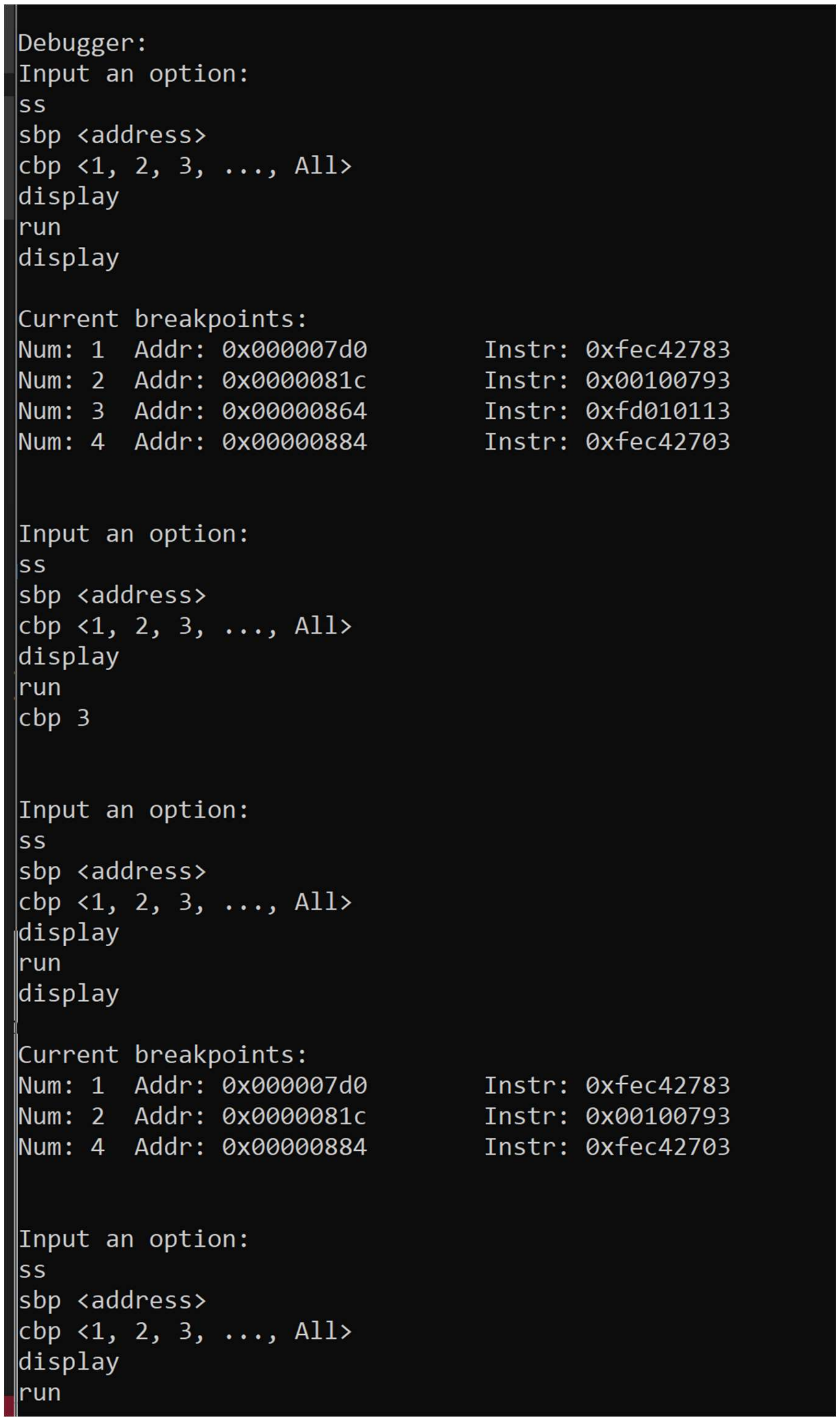
```
Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
run




Current PC address: 0x00000864

Current Register States:
        zero: 0x00000000          ra: 0x00000948          sp: 0x0007ffdc          gp: 0x00000000
          tp: 0x000010a2          t0: 0x000010a0          t1: 0x00000000          t2: 0x00000000
          s0: 0x0007fffc          s1: 0x00000000          a0: 0x00000006          a1: 0x00000000
          a2: 0x00000000          a3: 0x00000000          a4: 0x00000000          a5: 0x00000006
          a6: 0x00000000          a7: 0x00000000          s2: 0x00000000          s3: 0x00000000
          s4: 0x00000000          s5: 0x00000000          s6: 0x00000000          s7: 0x00000000
          s8: 0x00000000          s9: 0x00000000         s10: 0x00000000         s11: 0x00000000
          t3: 0x00000000          t4: 0x00000000          t5: 0x00000000          t6: 0x00000000

Currently executing line of code from file: HW05.cpp
{

Currently Executing Assembly Instruction(s):
        864:    fd010113                addi    sp,sp,-48
        868:    02812623                sw      s0,44(sp)
        86c:    03010413                addi    s0,sp,48
        870:    fca42e23                sw      a0,4060(s0)

Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
```

Single-Stepping to Next C Code Line



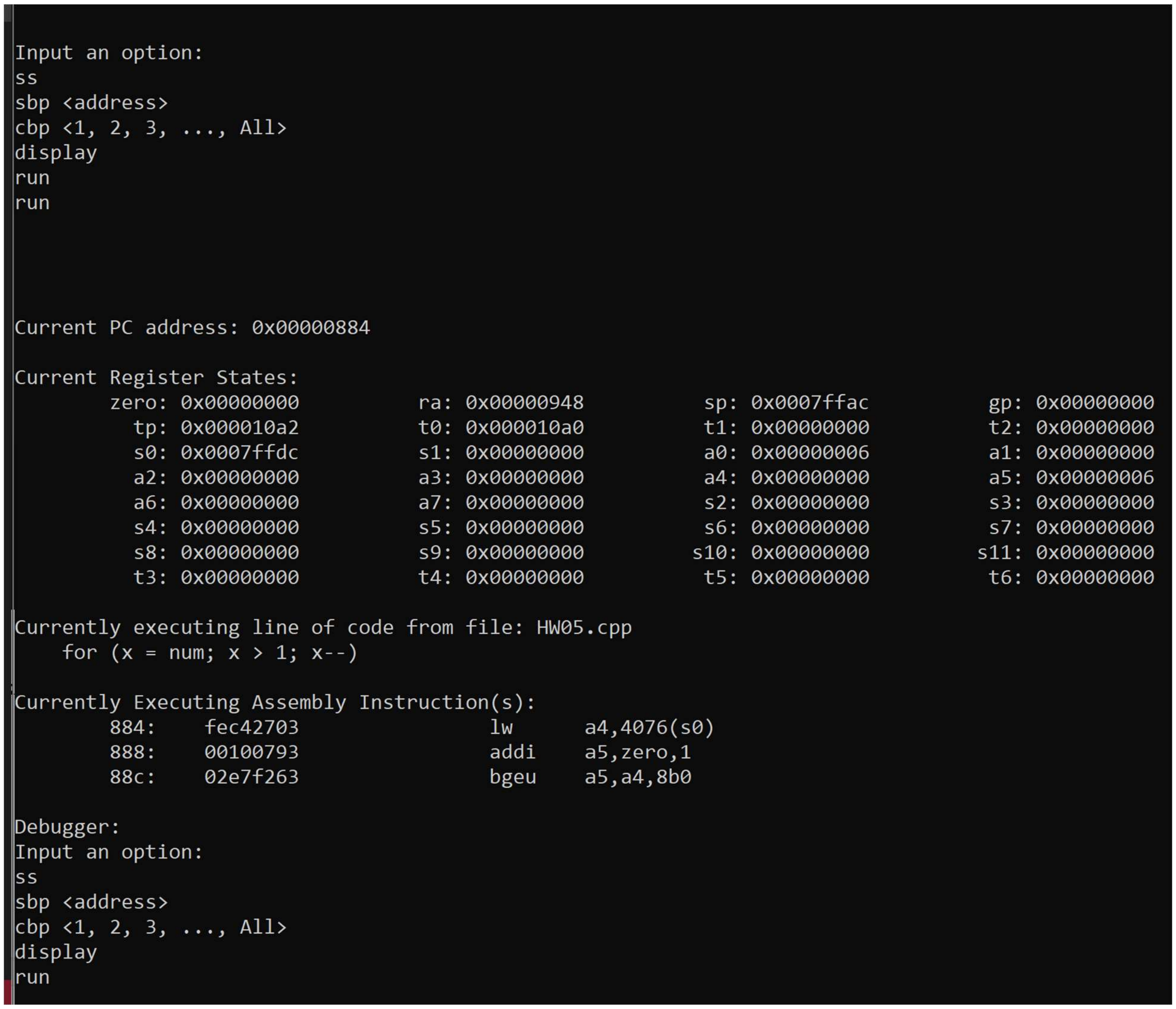
```
Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
ss




Current PC address: 0x00000874

Current Register States:
        zero: 0x00000000          ra: 0x00000948          sp: 0x0007ffac          gp: 0x00000000
          tp: 0x000010a2          t0: 0x000010a0          t1: 0x00000000          t2: 0x00000000
          s0: 0x0007ffdc          s1: 0x00000000          a0: 0x00000006          a1: 0x00000000
          a2: 0x00000000          a3: 0x00000000          a4: 0x00000000          a5: 0x00000006
          a6: 0x00000000          a7: 0x00000000          s2: 0x00000000          s3: 0x00000000
          s4: 0x00000000          s5: 0x00000000          s6: 0x00000000          s7: 0x00000000
          s8: 0x00000000          s9: 0x00000000         s10: 0x00000000         s11: 0x00000000
          t3: 0x00000000          t4: 0x00000000          t5: 0x00000000          t6: 0x00000000

Currently executing line of code from file: HW05.cpp
    uint32_t retVal = 1;

Currently Executing Assembly Instruction(s):
        874:    00100793                addi    a5,zero,1
        878:    fef42423                sw      a5,4072(s0)

Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
```

Clearing Current Breakpoint



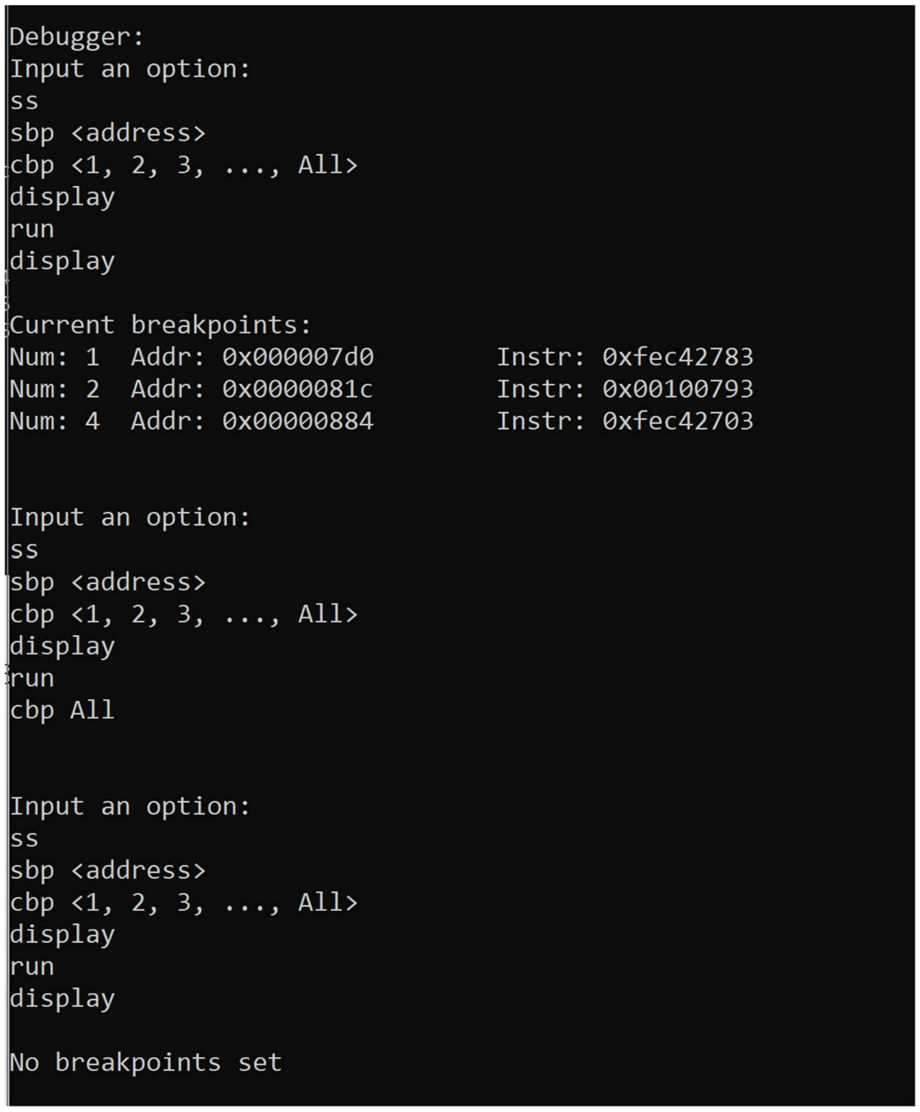
```
Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
display

Current breakpoints:
Num: 1  Addr: 0x000007d0        Instr: 0xfec42783
Num: 2  Addr: 0x0000081c        Instr: 0x00100793
Num: 3  Addr: 0x00000864        Instr: 0xfd010113
Num: 4  Addr: 0x00000884        Instr: 0xfec42703


Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
cbp 3


Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
display

Current breakpoints:
Num: 1  Addr: 0x000007d0        Instr: 0xfec42783
Num: 2  Addr: 0x0000081c        Instr: 0x00100793
Num: 4  Addr: 0x00000884        Instr: 0xfec42703


Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
```

Running to Next Breakpoint



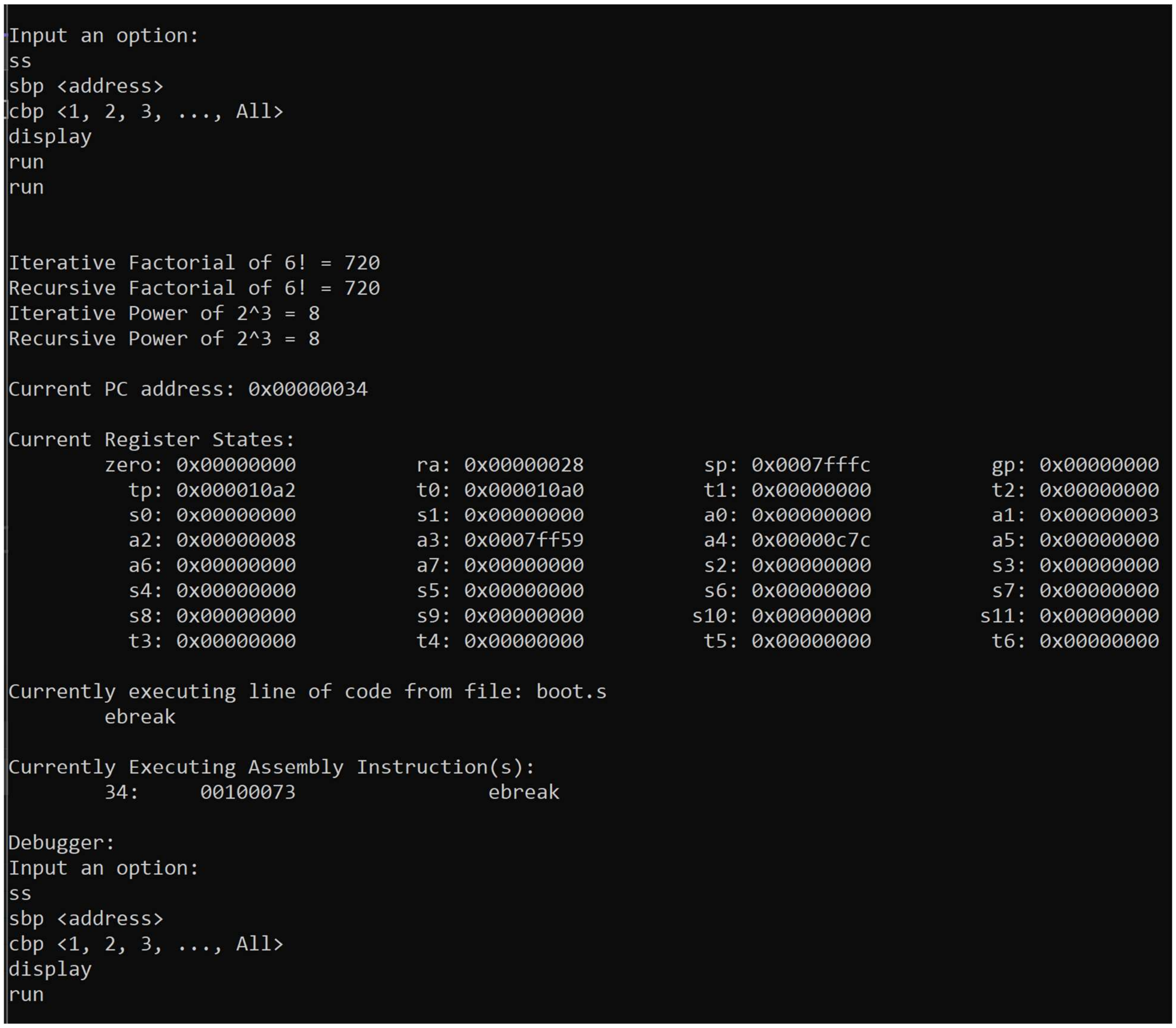
```
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
run




Current PC address: 0x00000884

Current Register States:
        zero: 0x00000000          ra: 0x00000948          sp: 0x0007ffac          gp: 0x00000000
          tp: 0x000010a2          t0: 0x000010a0          t1: 0x00000000          t2: 0x00000000
          s0: 0x0007ffdc          s1: 0x00000000          a0: 0x00000006          a1: 0x00000000
          a2: 0x00000000          a3: 0x00000000          a4: 0x00000000          a5: 0x00000006
          a6: 0x00000000          a7: 0x00000000          s2: 0x00000000          s3: 0x00000000
          s4: 0x00000000          s5: 0x00000000          s6: 0x00000000          s7: 0x00000000
          s8: 0x00000000          s9: 0x00000000         s10: 0x00000000         s11: 0x00000000
          t3: 0x00000000          t4: 0x00000000          t5: 0x00000000          t6: 0x00000000

Currently executing line of code from file: HW05.cpp
    for (x = num; x > 1; x--)

Currently Executing Assembly Instruction(s):
        884:    fec42703                lw      a4,4076(s0)
        888:    00100793                addi    a5,zero,1
        88c:    02e7f263                bgeu    a5,a4,8b0

Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
```

Clearing All Breakpoints



```
Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
display

Current breakpoints:
Num: 1  Addr: 0x000007d0        Instr: 0xfec42783
Num: 2  Addr: 0x0000081c        Instr: 0x00100793
Num: 4  Addr: 0x00000884        Instr: 0xfec42703


Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
cbp All


Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
display

No breakpoints set
```

Running Until End of Program (with Homework05 output)



```
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
run


Iterative Factorial of 6! = 720
Recursive Factorial of 6! = 720
Iterative Power of 2^3 = 8
Recursive Power of 2^3 = 8

Current PC address: 0x00000034

Current Register States:
        zero: 0x00000000          ra: 0x00000028          sp: 0x0007fffc          gp: 0x00000000
          tp: 0x000010a2          t0: 0x000010a0          t1: 0x00000000          t2: 0x00000000
          s0: 0x00000000          s1: 0x00000000          a0: 0x00000000          a1: 0x00000003
          a2: 0x00000008          a3: 0x0007ff59          a4: 0x00000c7c          a5: 0x00000000
          a6: 0x00000000          a7: 0x00000000          s2: 0x00000000          s3: 0x00000000
          s4: 0x00000000          s5: 0x00000000          s6: 0x00000000          s7: 0x00000000
          s8: 0x00000000          s9: 0x00000000         s10: 0x00000000         s11: 0x00000000
          t3: 0x00000000          t4: 0x00000000          t5: 0x00000000          t6: 0x00000000

Currently executing line of code from file: boot.s
        ebreak

Currently Executing Assembly Instruction(s):
        34:     00100073                ebreak

Debugger:
Input an option:
ss
sbp <address>
cbp <1, 2, 3, ..., All>
display
run
```