LAB SECTION COP 3402

Fall 2023

WEEK 3

INSTRUCTOR: Dr. Gary T. Leavens

GTA: Mana Mostaani

Simplified RISC Machine Manual

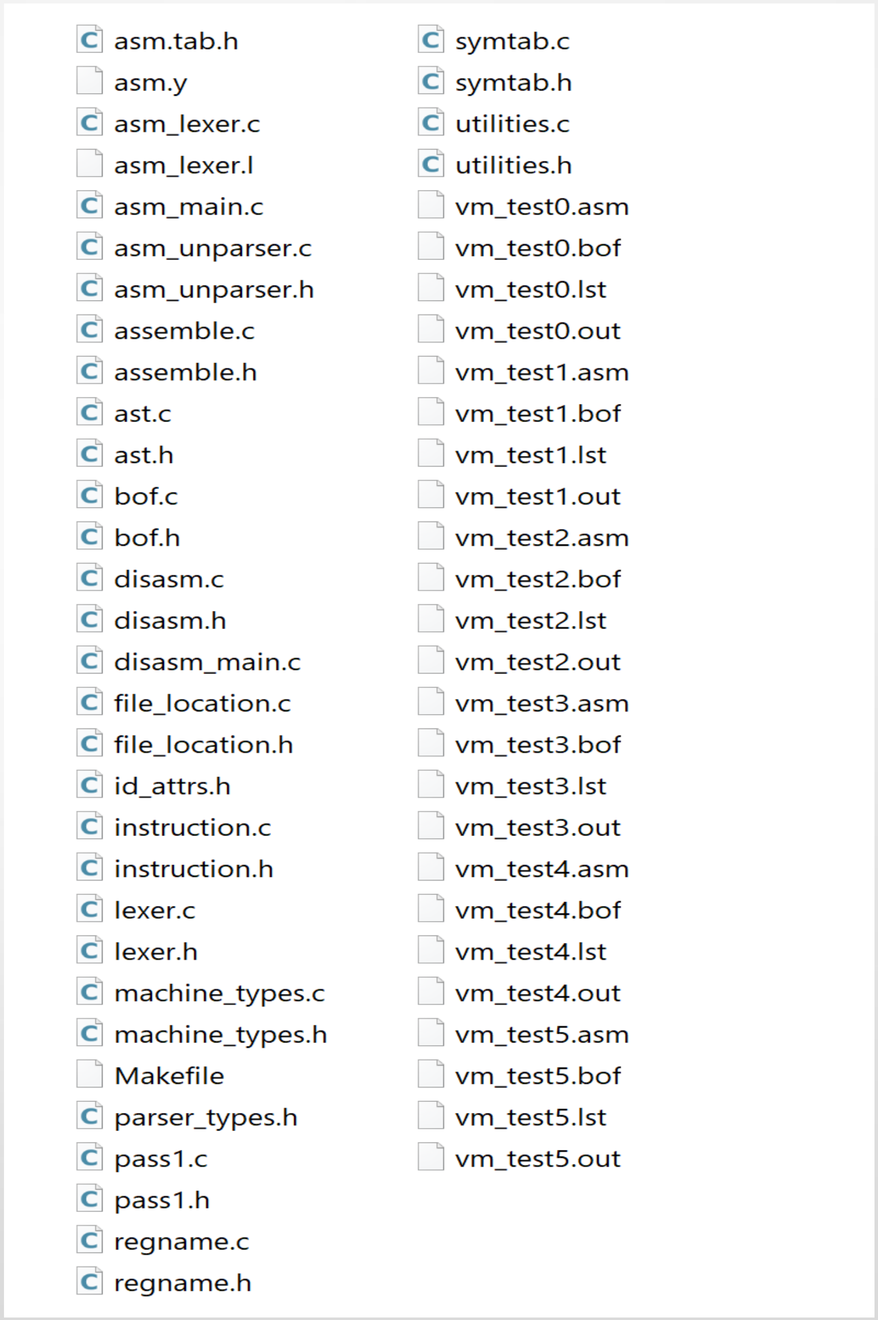
- **Goal:** Implementing a Virtual Machine taking a binary object file (.bof) and producing two outputs:
  - **.myo** (Tracing)
  - **.myp** (the output itself)

Hw1-tests

- **Assembler**
- **Makefile**

A Makefile is a script managing build systems for software projects

- **VM Test Cases**

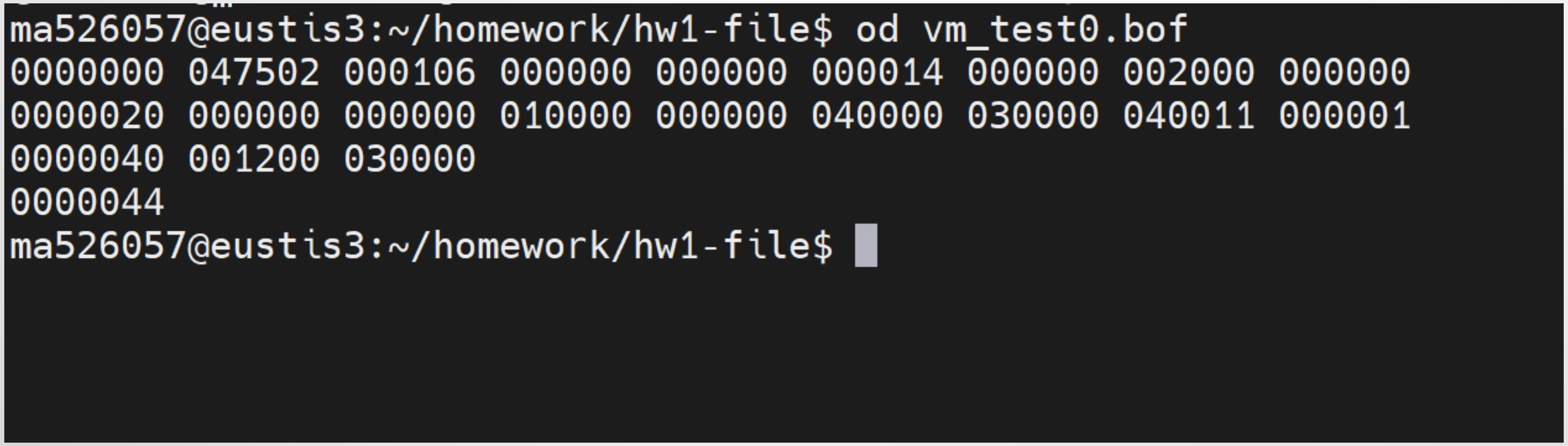


asm.tab.h
asm.y
asm_lexer.c
asm_lexer.l
asm_main.c
asm_unparser.c
asm_unparser.h
assemble.c
assemble.h
ast.c
ast.h
bof.c
bof.h
disasm.c
disasm.h
disasm_main.c
file_location.c
file_location.h
id_attrs.h
instruction.c
instruction.h
lexer.c
lexer.h
machine_types.c
machine_types.h
Makefile
parser_types.h
pass1.c
pass1.h
regname.c
regname.h
symtab.c
symtab.h
utilities.c
utilities.h
vm_test0.asm
vm_test0.bof
vm_test0.lst
vm_test0.out
vm_test1.asm
vm_test1.bof
vm_test1.lst
vm_test1.out
vm_test2.asm
vm_test2.bof
vm_test2.lst
vm_test2.out
vm_test3.asm
vm_test3.bof
vm_test3.lst
vm_test3.out
vm_test4.asm
vm_test4.bof
vm_test4.lst
vm_test4.out
vm_test5.asm
vm_test5.bof
vm_test5.lst
vm_test5.out

vm_test0.asm

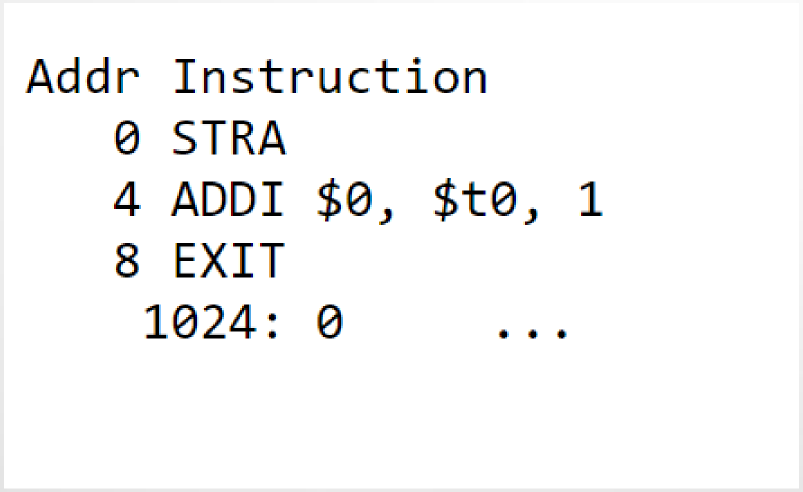
```
        # $Id: vm_test0.asm,v 1.1 2023/09/18 03:32:18 leavens Exp $
        .text start
start:  STRA
        ADDI $0, $t0, 1
        EXIT
        .data 1024
        .stack 4096
        .end
```

vm_test0.bof



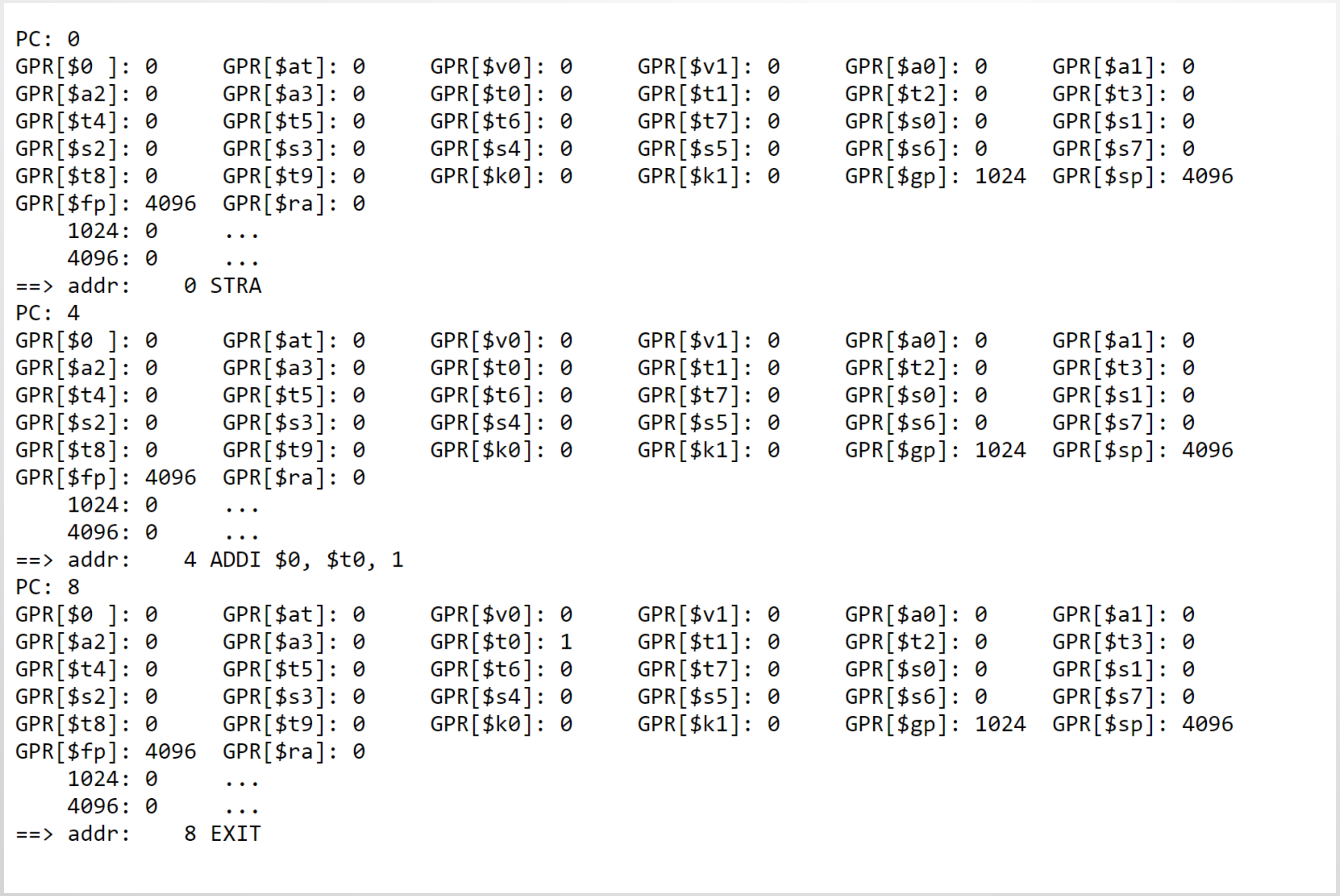
```
ma526057@eustis3:~/homework/hw1-file$ od vm_test0.bof
0000000 047502 000106 000000 000000 000014 000000 002000 000000
0000020 000000 000000 010000 000000 040000 030000 040011 000001
0000040 001200 030000
0000044
ma526057@eustis3:~/homework/hw1-file$
```

vm_test0.lst (.myp)



```
Addr Instruction
   0 STRA
   4 ADDI $0, $t0, 1
   8 EXIT
    1024: 0     ...
```

vm_test0.out (.myo)



```
PC: 0
GPR[$0 ]: 0     GPR[$at]: 0     GPR[$v0]: 0     GPR[$v1]: 0     GPR[$a0]: 0     GPR[$a1]: 0
GPR[$a2]: 0     GPR[$a3]: 0     GPR[$t0]: 0     GPR[$t1]: 0     GPR[$t2]: 0     GPR[$t3]: 0
GPR[$t4]: 0     GPR[$t5]: 0     GPR[$t6]: 0     GPR[$t7]: 0     GPR[$s0]: 0     GPR[$s1]: 0
GPR[$s2]: 0     GPR[$s3]: 0     GPR[$s4]: 0     GPR[$s5]: 0     GPR[$s6]: 0     GPR[$s7]: 0
GPR[$t8]: 0     GPR[$t9]: 0     GPR[$k0]: 0     GPR[$k1]: 0     GPR[$gp]: 1024  GPR[$sp]: 4096
GPR[$fp]: 4096  GPR[$ra]: 0
    1024: 0     ...
    4096: 0     ...
==> addr:    0 STRA
PC: 4
GPR[$0 ]: 0     GPR[$at]: 0     GPR[$v0]: 0     GPR[$v1]: 0     GPR[$a0]: 0     GPR[$a1]: 0
GPR[$a2]: 0     GPR[$a3]: 0     GPR[$t0]: 0     GPR[$t1]: 0     GPR[$t2]: 0     GPR[$t3]: 0
GPR[$t4]: 0     GPR[$t5]: 0     GPR[$t6]: 0     GPR[$t7]: 0     GPR[$s0]: 0     GPR[$s1]: 0
GPR[$s2]: 0     GPR[$s3]: 0     GPR[$s4]: 0     GPR[$s5]: 0     GPR[$s6]: 0     GPR[$s7]: 0
GPR[$t8]: 0     GPR[$t9]: 0     GPR[$k0]: 0     GPR[$k1]: 0     GPR[$gp]: 1024  GPR[$sp]: 4096
GPR[$fp]: 4096  GPR[$ra]: 0
    1024: 0     ...
    4096: 0     ...
==> addr:    4 ADDI $0, $t0, 1
PC: 8
GPR[$0 ]: 0     GPR[$at]: 0     GPR[$v0]: 0     GPR[$v1]: 0     GPR[$a0]: 0     GPR[$a1]: 0
GPR[$a2]: 0     GPR[$a3]: 0     GPR[$t0]: 1     GPR[$t1]: 0     GPR[$t2]: 0     GPR[$t3]: 0
GPR[$t4]: 0     GPR[$t5]: 0     GPR[$t6]: 0     GPR[$t7]: 0     GPR[$s0]: 0     GPR[$s1]: 0
GPR[$s2]: 0     GPR[$s3]: 0     GPR[$s4]: 0     GPR[$s5]: 0     GPR[$s6]: 0     GPR[$s7]: 0
GPR[$t8]: 0     GPR[$t9]: 0     GPR[$k0]: 0     GPR[$k1]: 0     GPR[$gp]: 1024  GPR[$sp]: 4096
GPR[$fp]: 4096  GPR[$ra]: 0
    1024: 0     ...
    4096: 0     ...
==> addr:    8 EXIT
```

Running the code

- All the files + test cases should be in the current directory
- Commands you need to run the code:
  - **make**

    Complies ASM and VM together (it does not work if there is no VM)
  - **make asm**

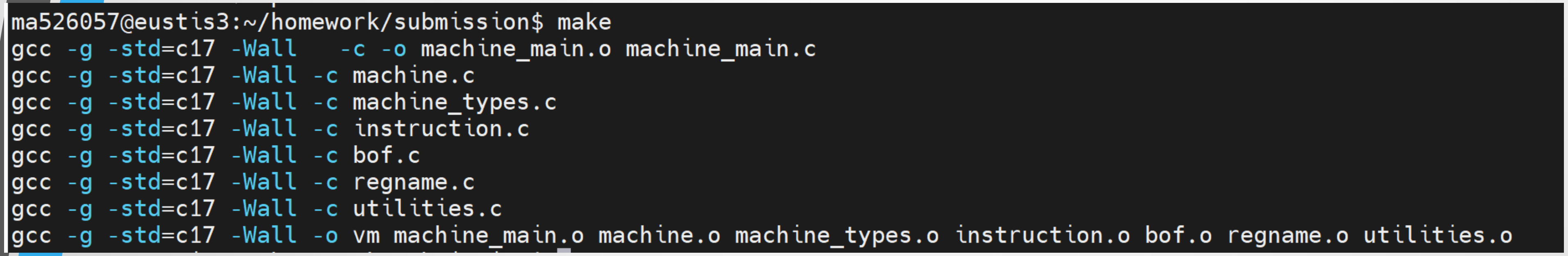
    Builds the assembler
  - **./asm vm_test0.asm**

    Runs the ASM and produces vm_test0.bof
  - **./vm vm_test0.bof**

    Runs the VM and prints the tracing
  - **./vm –p vm_test0.bof**

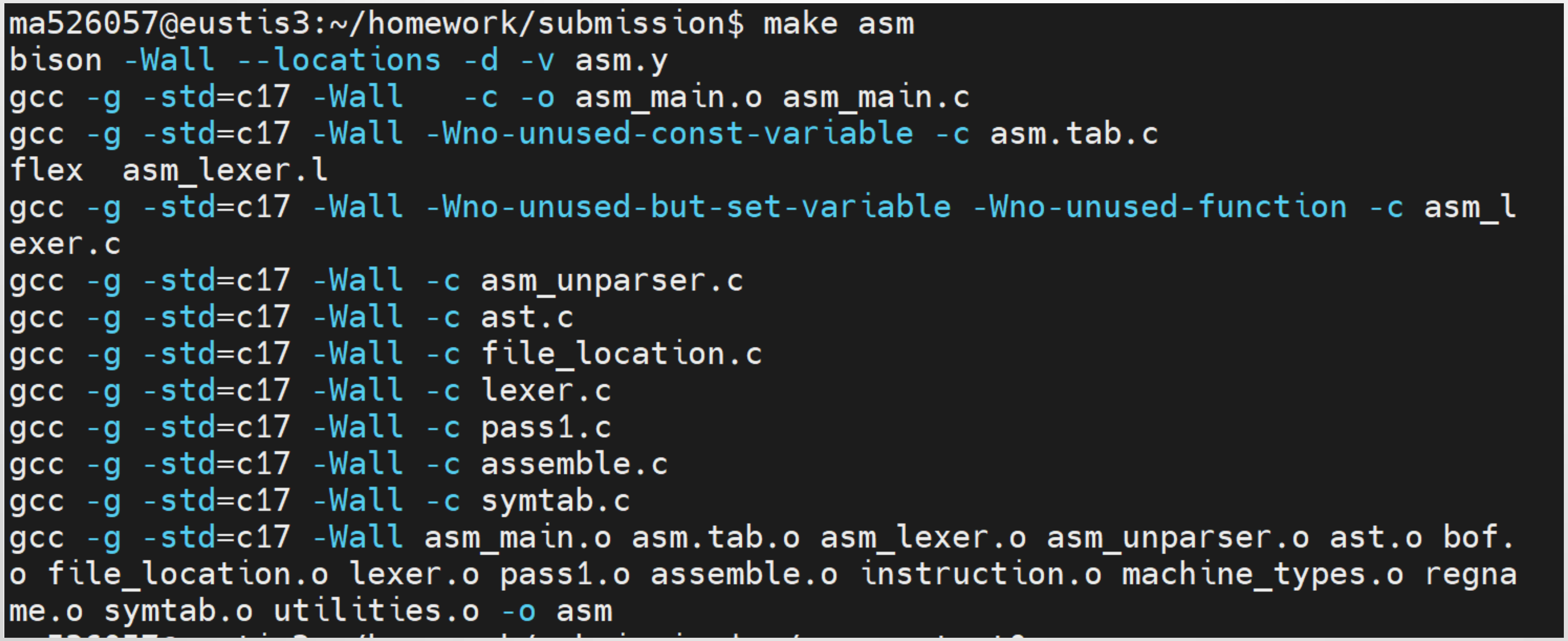
    Prints out the output file

make



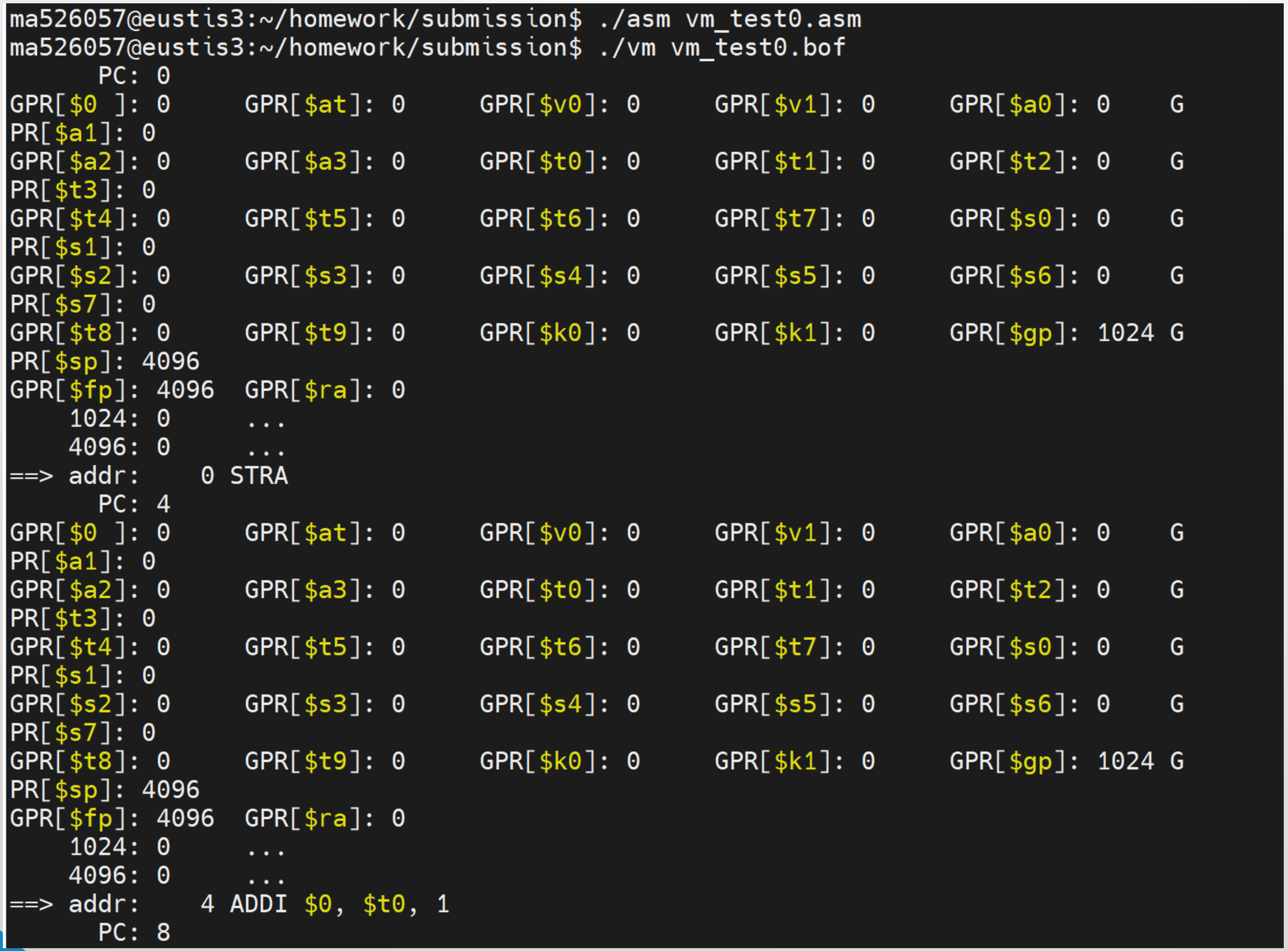
```
ma526057@eustis3:~/homework/submission$ make
gcc -g -std=c17 -Wall   -c -o machine_main.o machine_main.c
gcc -g -std=c17 -Wall -c machine.c
gcc -g -std=c17 -Wall -c machine_types.c
gcc -g -std=c17 -Wall -c instruction.c
gcc -g -std=c17 -Wall -c bof.c
gcc -g -std=c17 -Wall -c regname.c
gcc -g -std=c17 -Wall -c utilities.c
gcc -g -std=c17 -Wall -o vm machine_main.o machine.o machine_types.o instruction.o bof.o regname.o utilities.o
```

make asm



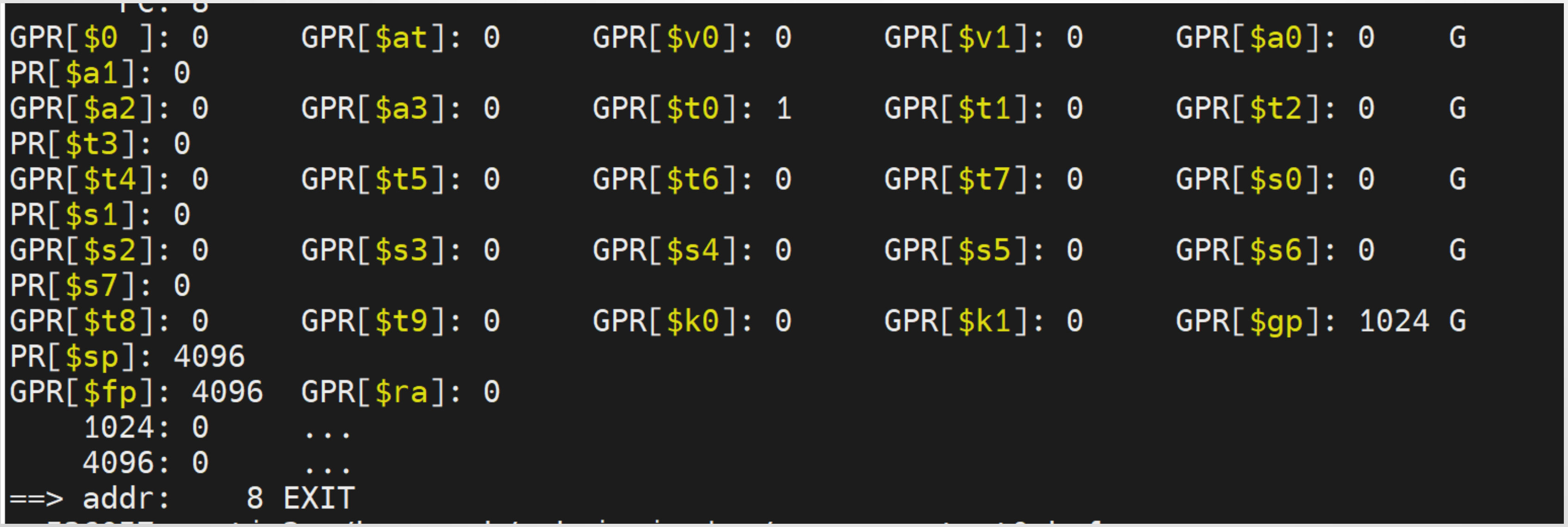
```
ma526057@eustis3:~/homework/submission$ make asm
bison -Wall --locations -d -v asm.y
gcc -g -std=c17 -Wall   -c -o asm_main.o asm_main.c
gcc -g -std=c17 -Wall -Wno-unused-const-variable -c asm.tab.c
flex  asm_lexer.l
gcc -g -std=c17 -Wall -Wno-unused-but-set-variable -Wno-unused-function -c asm_l
exer.c
gcc -g -std=c17 -Wall -c asm_unparser.c
gcc -g -std=c17 -Wall -c ast.c
gcc -g -std=c17 -Wall -c file_location.c
gcc -g -std=c17 -Wall -c lexer.c
gcc -g -std=c17 -Wall -c pass1.c
gcc -g -std=c17 -Wall -c assemble.c
gcc -g -std=c17 -Wall -c symtab.c
gcc -g -std=c17 -Wall asm_main.o asm.tab.o asm_lexer.o asm_unparser.o ast.o bof.
o file_location.o lexer.o pass1.o assemble.o instruction.o machine_types.o regna
me.o symtab.o utilities.o -o asm
```

./asm vm_test0.asm ./vm vm_test0.bof



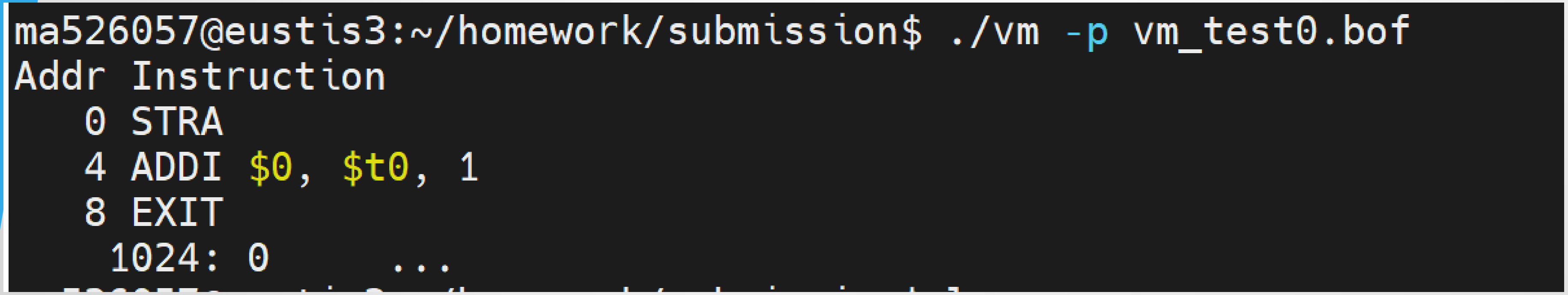
```
ma526057@eustis3:~/homework/submission$ ./asm vm_test0.asm
ma526057@eustis3:~/homework/submission$ ./vm vm_test0.bof
      PC: 0
GPR[$0 ]: 0     GPR[$at]: 0     GPR[$v0]: 0     GPR[$v1]: 0     GPR[$a0]: 0    G
PR[$a1]: 0
GPR[$a2]: 0     GPR[$a3]: 0     GPR[$t0]: 0     GPR[$t1]: 0     GPR[$t2]: 0    G
PR[$t3]: 0
GPR[$t4]: 0     GPR[$t5]: 0     GPR[$t6]: 0     GPR[$t7]: 0     GPR[$s0]: 0    G
PR[$s1]: 0
GPR[$s2]: 0     GPR[$s3]: 0     GPR[$s4]: 0     GPR[$s5]: 0     GPR[$s6]: 0    G
PR[$s7]: 0
GPR[$t8]: 0     GPR[$t9]: 0     GPR[$k0]: 0     GPR[$k1]: 0     GPR[$gp]: 1024 G
PR[$sp]: 4096
GPR[$fp]: 4096  GPR[$ra]: 0
    1024: 0     ...
    4096: 0     ...
==> addr:    0 STRA
      PC: 4
GPR[$0 ]: 0     GPR[$at]: 0     GPR[$v0]: 0     GPR[$v1]: 0     GPR[$a0]: 0    G
PR[$a1]: 0
GPR[$a2]: 0     GPR[$a3]: 0     GPR[$t0]: 0     GPR[$t1]: 0     GPR[$t2]: 0    G
PR[$t3]: 0
GPR[$t4]: 0     GPR[$t5]: 0     GPR[$t6]: 0     GPR[$t7]: 0     GPR[$s0]: 0    G
PR[$s1]: 0
GPR[$s2]: 0     GPR[$s3]: 0     GPR[$s4]: 0     GPR[$s5]: 0     GPR[$s6]: 0    G
PR[$s7]: 0
GPR[$t8]: 0     GPR[$t9]: 0     GPR[$k0]: 0     GPR[$k1]: 0     GPR[$gp]: 1024 G
PR[$sp]: 4096
GPR[$fp]: 4096  GPR[$ra]: 0
    1024: 0     ...
    4096: 0     ...
==> addr:    4 ADDI $0, $t0, 1
      PC: 8
```

./asm vm_test0.asm
./vm vm_test0.bof



```
GPR[$0 ]: 0     GPR[$at]: 0     GPR[$v0]: 0     GPR[$v1]: 0     GPR[$a0]: 0    G
PR[$a1]: 0
GPR[$a2]: 0     GPR[$a3]: 0     GPR[$t0]: 1     GPR[$t1]: 0     GPR[$t2]: 0    G
PR[$t3]: 0
GPR[$t4]: 0     GPR[$t5]: 0     GPR[$t6]: 0     GPR[$t7]: 0     GPR[$s0]: 0    G
PR[$s1]: 0
GPR[$s2]: 0     GPR[$s3]: 0     GPR[$s4]: 0     GPR[$s5]: 0     GPR[$s6]: 0    G
PR[$s7]: 0
GPR[$t8]: 0     GPR[$t9]: 0     GPR[$k0]: 0     GPR[$k1]: 0     GPR[$gp]: 1024 G
PR[$sp]: 4096
GPR[$fp]: 4096  GPR[$ra]: 0
    1024: 0     ...
    4096: 0     ...
==> addr:    8 EXIT
```

./vm –p vm_test0.bof



```
ma526057@eustis3:~/homework/submission$ ./vm -p vm_test0.bof
Addr Instruction
   0 STRA
   4 ADDI $0, $t0, 1
   8 EXIT
    1024: 0     ...
```

Other Useful Commands

- **make vm_test0.myo**

Creates vm_test0.myo file

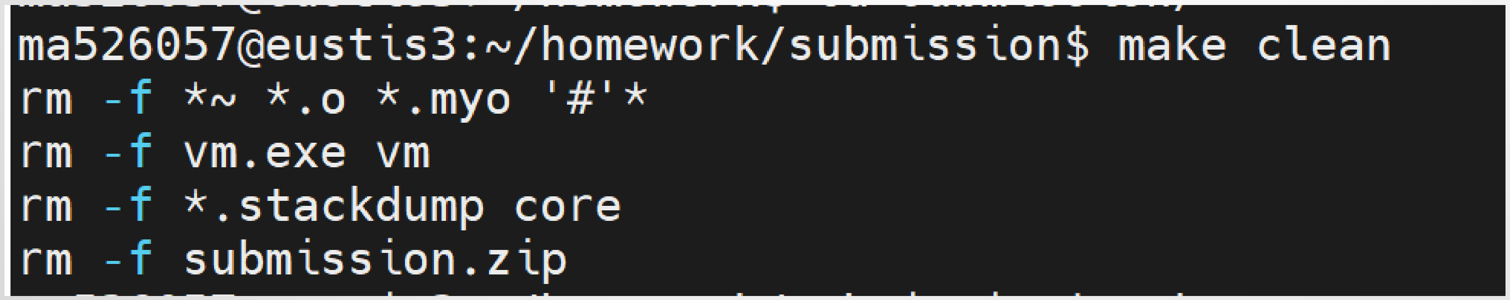
- **make vm_test0.myp**

Creates vm_test0.myp file

- **make clean**

Cleans up the directory

make clean

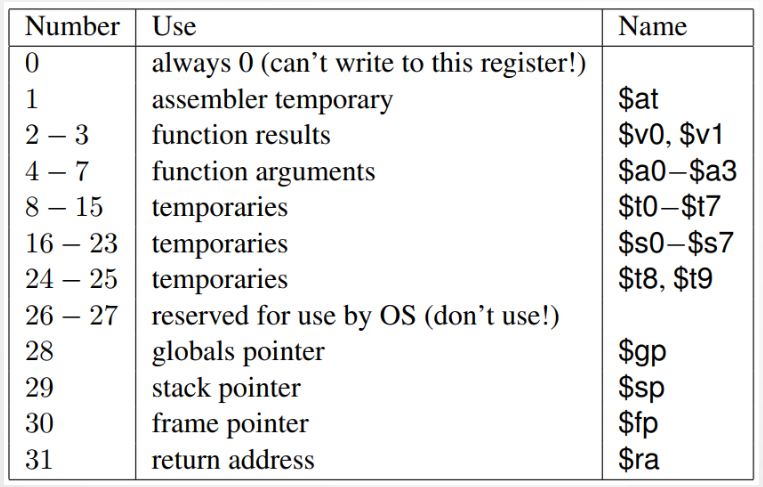


```
ma526057@eustis3:~/homework/submission$ make clean
rm -f *~ *.o *.myo '#'*
rm -f vm.exe vm
rm -f *.stackdump core
rm -f submission.zip
```

SRM

SRM is register machine having:

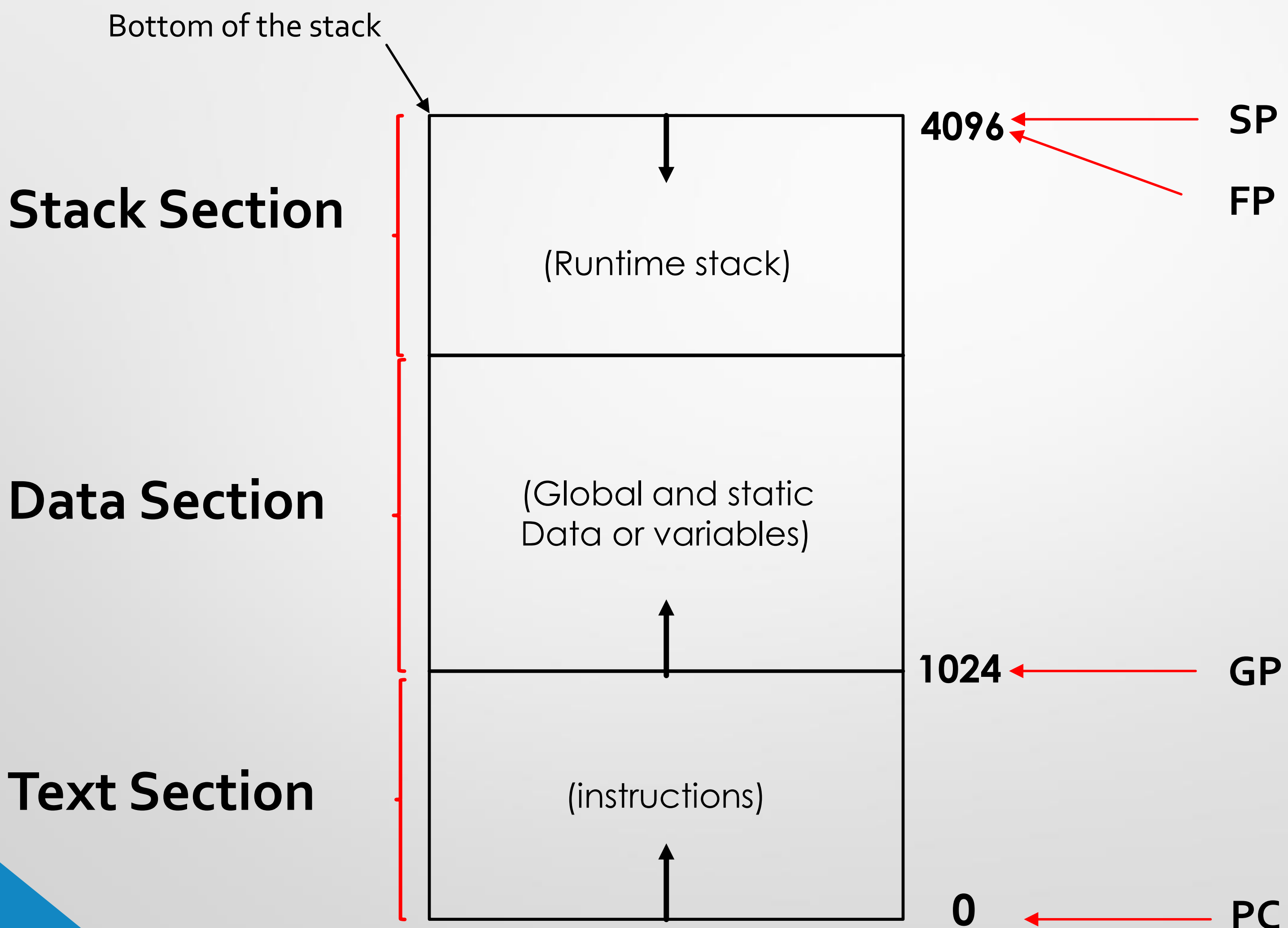
- **32 32-bit General Purpose Registers (GPR)**



Number	Use	Name
0	always 0 (can't write to this register!)	
1	assembler temporary	\$at
2 – 3	function results	\$v0, \$v1
4 – 7	function arguments	\$a0–\$a3
8 – 15	temporaries	\$t0–\$t7
16 – 23	temporaries	\$s0–\$s7
24 – 25	temporaries	\$t8, \$t9
26 – 27	reserved for use by OS (don't use!)	
28	globals pointer	\$gp
29	stack pointer	\$sp
30	frame pointer	\$fp
31	return address	\$ra

- **Special Purpose Registers**
  - **PC:** address of the next instruction to execute
  - **HI:** most significant bits of the result of a multiplication or the remainder in a division
  - **LO:** least significant bits of the result of a multiplication or the quotient in a division

Memory


Bottom of the stack
Stack Section
(Runtime stack)
4096
SP
FP
Data Section
(Global and static Data or variables)
1024
GP
Text Section
(instructions)
0
PC

SRM's Assembly Language

- **.text <entry-point>**
  - The address of the program's entry point (PC)
  - It might be a number or a label
  - The beginning of the text section
- **.data <static-start-address>**
  - The value is the initial value of the $gp register
  - The beginning of the data section
- **.stack <stack-bottom-address>**
  - The address of the bottom of the stack
  - The initial values of $sp and $fp are the same
  - $sp points to the address of the top of the stack
  - ** Current AR(Activation Record) is between FP and SP**

Attendance Code

Email's Subject: WEEK 3 ATTENDANCE

Attendance Code: 56830

Email: Mana.Mostaani@ucf.edu