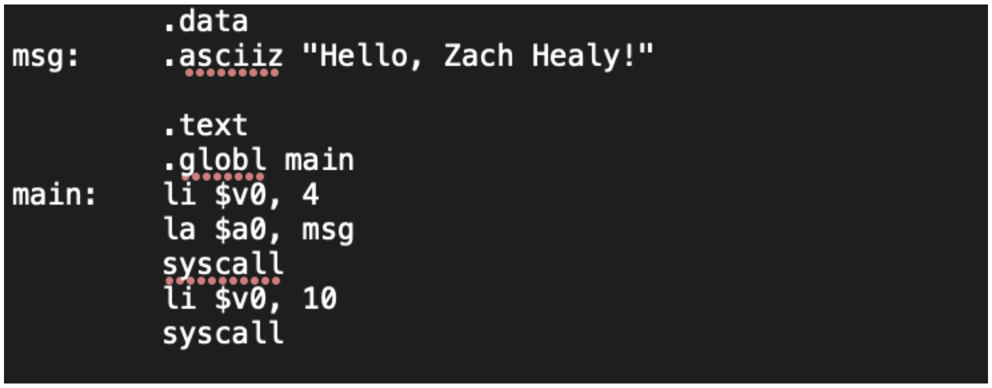
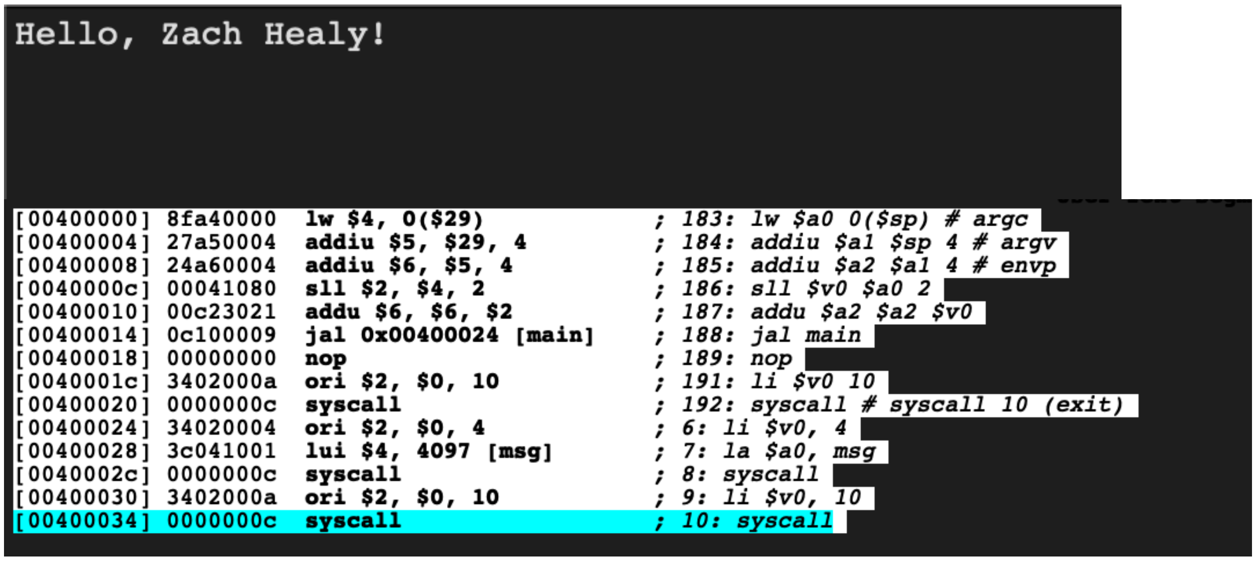
1.



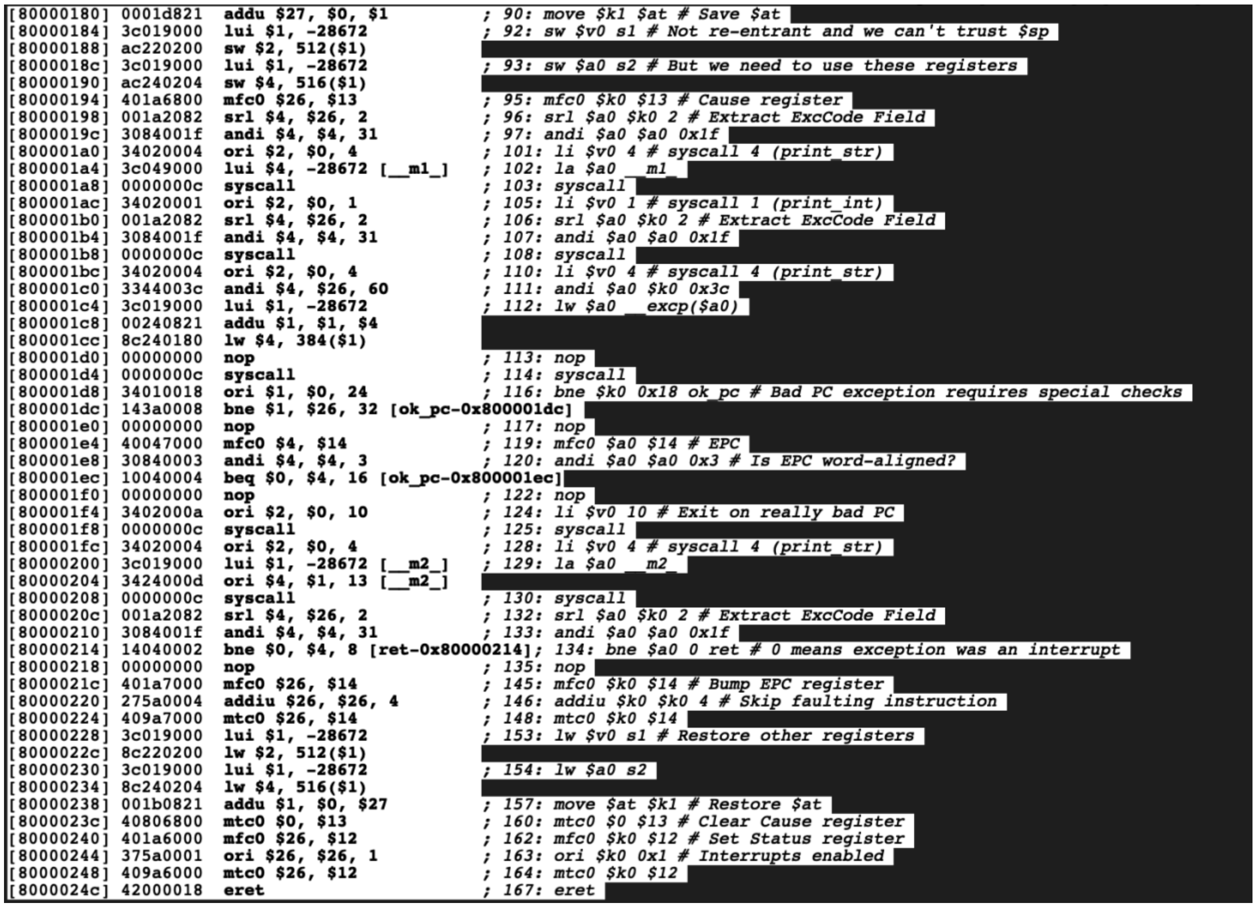
```
        .data
msg:    .asciiz "Hello, Zach Healy!"

        .text
        .globl main
main:   li $v0, 4
        la $a0, msg
        syscall
        li $v0, 10
        syscall
```



```
Hello, Zach Healy!
```

```
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4              ; 6: li $v0, 4
[00400028] 3c041001  lui $4, 4097 [msg]         ; 7: la $a0, msg
[0040002c] 0000000c  syscall                    ; 8: syscall
[00400030] 3402000a  ori $2, $0, 10             ; 9: li $v0, 10
[00400034] 0000000c  syscall                    ; 10: syscall
```



```
[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13              ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2             ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31            ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4              ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]     ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                    ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1              ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2             ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31            ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                    ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4              ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60           ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672             ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
[800001d0] 00000000  nop                        ; 113: nop
[800001d4] 0000000c  syscall                    ; 114: syscall
[800001d8] 34010018  ori $1, $0, 24             ; 116: bne $k0 0x18 ok_pc # Bad PC exception requires special checks
[800001dc] 143a0008  bne $1, $26, 32 [ok_pc-0x800001dc]
[800001e0] 00000000  nop                        ; 117: nop
[800001e4] 40047000  mfc0 $4, $14               ; 119: mfc0 $a0 $14 # EPC
[800001e8] 30840003  andi $4, $4, 3             ; 120: andi $a0 $a0 0x3 # Is EPC word-aligned?
[800001ec] 10040004  beq $0, $4, 16 [ok_pc-0x800001ec]
[800001f0] 00000000  nop                        ; 122: nop
[800001f4] 3402000a  ori $2, $0, 10             ; 124: li $v0 10 # Exit on really bad PC
[800001f8] 0000000c  syscall                    ; 125: syscall
[800001fc] 34020004  ori $2, $0, 4              ; 128: li $v0 4 # syscall 4 (print_str)
[80000200] 3c019000  lui $1, -28672 [__m2_]     ; 129: la $a0 __m2_
[80000204] 3424000d  ori $4, $1, 13 [__m2_]
[80000208] 0000000c  syscall                    ; 130: syscall
[8000020c] 001a2082  srl $4, $26, 2             ; 132: srl $a0 $k0 2 # Extract ExcCode Field
[80000210] 3084001f  andi $4, $4, 31            ; 133: andi $a0 $a0 0x1f
[80000214] 14040002  bne $0, $4, 8 [ret-0x80000214]; 134: bne $a0 0 ret # 0 means exception was an interrupt
[80000218] 00000000  nop                        ; 135: nop
[8000021c] 401a7000  mfc0 $26, $14              ; 145: mfc0 $k0 $14 # Bump EPC register
[80000220] 275a0004  addiu $26, $26, 4          ; 146: addiu $k0 $k0 4 # Skip faulting instruction
[80000224] 409a7000  mtc0 $26, $14              ; 148: mtc0 $k0 $14
[80000228] 3c019000  lui $1, -28672             ; 153: lw $v0 s1 # Restore other registers
[8000022c] 8c220200  lw $2, 512($1)
[80000230] 3c019000  lui $1, -28672             ; 154: lw $a0 s2
[80000234] 8c240204  lw $4, 516($1)
[80000238] 001b0821  addu $1, $0, $27           ; 157: move $at $k1 # Restore $at
[8000023c] 40806800  mtc0 $0, $13               ; 160: mtc0 $0 $13 # Clear Cause register
[80000240] 401a6000  mfc0 $26, $12              ; 162: mfc0 $k0 $12 # Set Status register
[80000244] 375a0001  ori $26, $26, 1            ; 163: ori $k0 0x1 # Interrupts enabled
[80000248] 409a6000  mtc0 $26, $12              ; 164: mtc0 $k0 $12
[8000024c] 42000018  eret                       ; 167: eret
```

2. Things that changed
   a. PC = 400034
      i. this is a program counter that increments in 4s to help point to a part of the program.
   b. R2 [v0] = a
      i. V0 is used as a way of setting a value for a system call.
   c. R4 [a0] = 10010000
      i. A0 stores the value of whatever is being called.
   d. R31 [ra] = 400018
      i. This is the return address that can be used to save and restore addresses during a call function