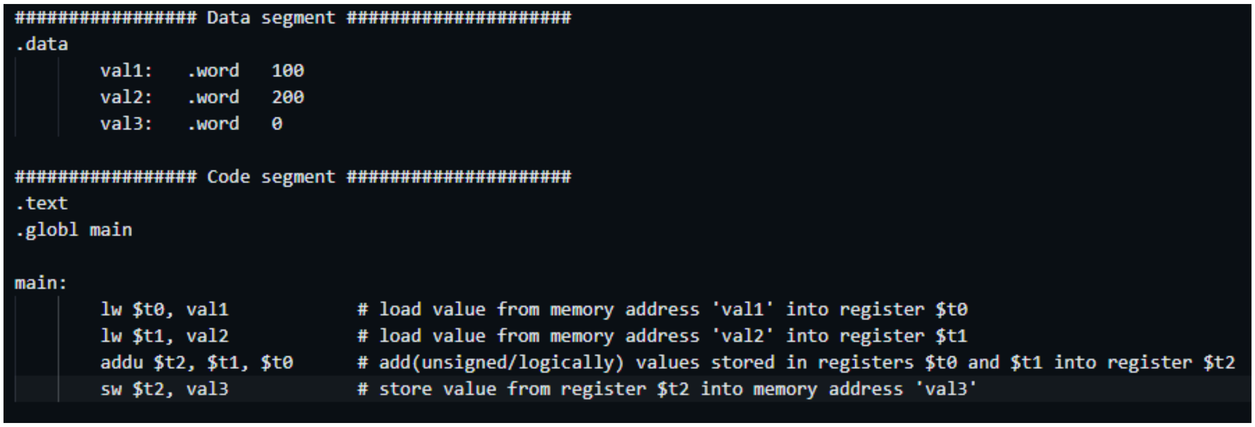
Assignment A P1-1 Assembly Program Lab 1. Sum of two values

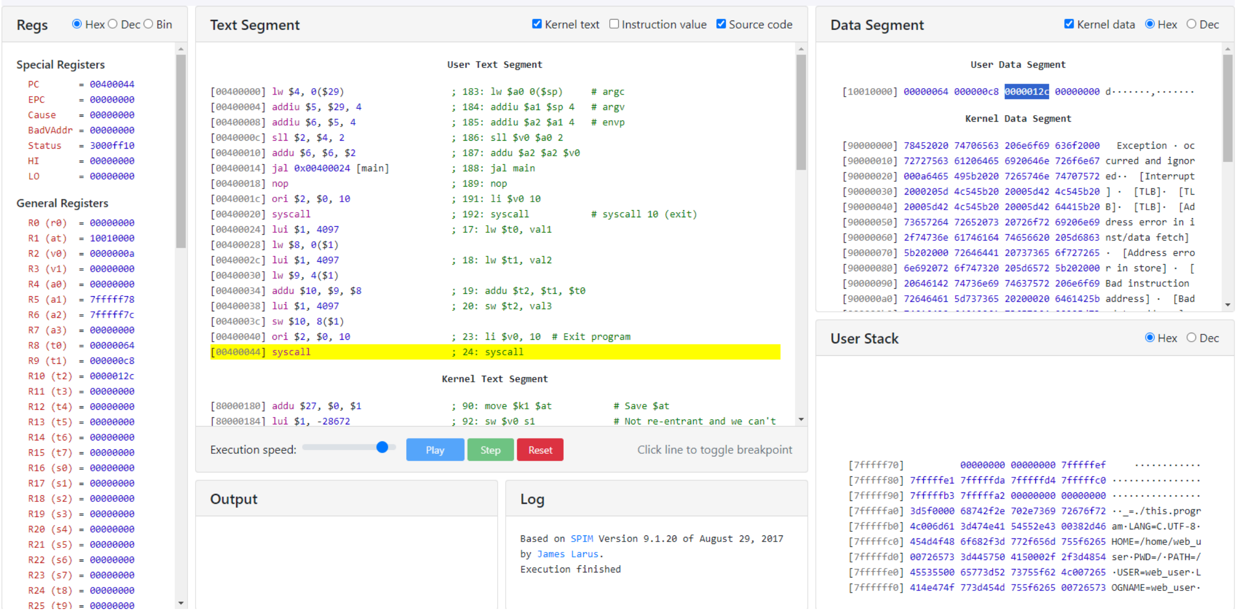
Submit Items:
GitHub Link to your file directly. (Please make it a link, not a text)
https://github.com/AVC-CS/p1a1-mtvonbargen/blob/2e8a28612e534a9d47ab0717966f8b5d4d0cdc4a/P1A1.asm

Elaborate on your program



```
################# Data segment #####################
.data
        val1:   .word   100
        val2:   .word   200
        val3:   .word   0

################# Code segment #####################
.text
.globl main

main:
        lw $t0, val1            # load value from memory address 'val1' into register $t0
        lw $t1, val2            # load value from memory address 'val2' into register $t1
        addu $t2, $t1, $t0      # add(unsigned/logically) values stored in registers $t0 and $t1 into register $t2
        sw $t2, val3            # store value from register $t2 into memory address 'val3'
```

Submit the run result(with register value)



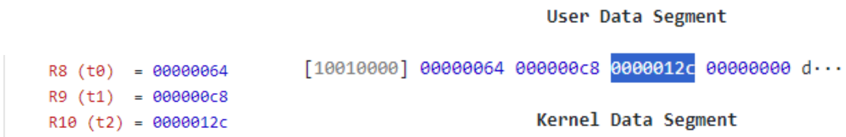
```
Special Registers
  PC       = 00400044
  EPC      = 00000000
  Cause    = 00000000
  BadVAddr = 00000000
  Status   = 3000ff10
  HI       = 00000000
  LO       = 00000000

General Registers
  R0 (r0)  = 00000000
  R1 (at)  = 10010000
  R2 (v0)  = 0000000a
  R3 (v1)  = 00000000
  R4 (a0)  = 00000000
  R5 (a1)  = 7fffff78
  R6 (a2)  = 7fffff7c
  R7 (a3)  = 00000000
  R8 (t0)  = 00000064
  R9 (t1)  = 000000c8
  R10 (t2) = 0000012c
  R11 (t3) = 00000000
  R12 (t4) = 00000000
  R13 (t5) = 00000000
  R14 (t6) = 00000000
  R15 (t7) = 00000000
  R16 (s0) = 00000000
  R17 (s1) = 00000000
  R18 (s2) = 00000000
  R19 (s3) = 00000000
  R20 (s4) = 00000000
  R21 (s5) = 00000000
  R22 (s6) = 00000000
  R23 (s7) = 00000000
  R24 (t8) = 00000000
  R25 (t9) = 00000000
```

```
                         User Text Segment

[00400000] lw $4, 0($29)               ; 183: lw $a0 0($sp)     # argc
[00400004] addiu $5, $29, 4            ; 184: addiu $a1 $sp 4   # argv
[00400008] addiu $6, $5, 4             ; 185: addiu $a2 $a1 4   # envp
[0040000c] sll $2, $4, 2               ; 186: sll $v0 $a0 2
[00400010] addu $6, $6, $2             ; 187: addu $a2 $a2 $v0
[00400014] jal 0x00400024 [main]       ; 188: jal main
[00400018] nop                         ; 189: nop
[0040001c] ori $2, $0, 10              ; 191: li $v0 10
[00400020] syscall                     ; 192: syscall           # syscall 10 (exit)
[00400024] lui $1, 4097                ; 17: lw $t0, val1
[00400028] lw $8, 0($1)
[0040002c] lui $1, 4097                ; 18: lw $t1, val2
[00400030] lw $9, 4($1)
[00400034] addu $10, $9, $8            ; 19: addu $t2, $t1, $t0
[00400038] lui $1, 4097                ; 20: sw $t2, val3
[0040003c] sw $10, 8($1)
[00400040] ori $2, $0, 10              ; 23: li $v0, 10  # Exit program
[00400044] syscall                     ; 24: syscall

                         Kernel Text Segment

[80000180] addu $27, $0, $1            ; 90: move $k1 $at          # Save $at
[80000184] lui $1, -28672              ; 92: sw $v0 s1             # Not re-entrant and we can't
```

```
                    User Data Segment

[10010000] 00000064 000000c8 0000012c 00000000 d·······,·······

                    Kernel Data Segment

[90000000] 78452020 74706563 206e6f69 636f2000   Exception · oc
[90000010] 72727563 61206465 6920646e 726f6e67 curred and ignor
[90000020] 000a6465 495b2020 7265746e 74707572 ed··  [Interrupt
[90000030] 2000205d 4c545b20 20005d42 4c545b20 ] ·  [TLB]·  [TL
[90000040] 20005d42 4c545b20 20005d42 64415b20 B]·  [TLB]·  [Ad
[90000050] 73657264 72652073 20726f72 69206e69 dress error in i
[90000060] 2f74736e 61746164 74656620 205d6863 nst/data fetch]
[90000070] 5b202000 72646441 20737365 6f727265 ·  [Address erro
[90000080] 6e692072 6f747320 205d6572 5b202000 r in store] ·  [
[90000090] 20646142 74736e69 74637572 206e6f69 Bad instruction
[900000a0] 72646461 5d737365 20200020 6461425b address] ·  [Bad
```

```
[7fffff70]          00000000 00000000 7fffffef    ············
[7fffff80] 7fffffe1 7fffffda 7fffffd4 7fffffc0 ················
[7fffff90] 7fffffb3 7fffffa2 00000000 00000000 ················
[7fffffa0] 3d5f0000 68742f2e 702e7369 72676f72 ··_=./this.progr
[7fffffb0] 4c006d61 3d474e41 54552e43 00382d46 am·LANG=C.UTF-8·
[7fffffc0] 454d4f48 6f682f3d 772f656d 755f6265 HOME=/home/web_u
[7fffffd0] 00726573 3d445750 4150002f 2f3d4854 ser·PWD=/·PATH=/
[7fffffe0] 45535500 65773d52 73755f62 4c007265 ·USER=web_user·L
[7ffffff0] 414e474f 773d454d 755f6265 00726573 OGNAME=web_user·
```

```
Based on SPIM Version 9.1.20 of August 29, 2017
by James Larus.
Execution finished
```



```
                      User Data Segment

R8 (t0)  = 00000064    [10010000] 00000064 000000c8 0000012c 00000000 d···
R9 (t1)  = 000000c8
R10 (t2) = 0000012c           Kernel Data Segment
```

00000064 = 100
000000c8 = 200
0000012c = 300

Register Value. What is the purpose of this register? How is it updated throughout your program?

$t0 = to load the value stored in val1 for future use. It’s updated in the line **lw $t0, val1** which specifies $t0 as the destination register, and specifies which memory address to load the value from which is val1.

$t1 = to load the value stored in val2 for future use. It’s updated in the line **lw $t1, val2** which specifies $t1 as the destination register, and specifies which memory address to load the value from which is val2.

$t2 = to load the value produced by the addu function. It’s updated in the line **addu $t2, $t1, $t0** which specifies $t2 as the destination register, and specifies which registers to add logically/unsigned with, which is $t1 and $t0.

The interaction between register and memory space

The memory spaces are val1, val2, and val3.

Val1 stores the value ‘100’

Val2 stores the value ‘200’

And Val3 stores the sum of Val1 and Val 2.

Registers allow us to load the values so that they’re able to be instructed upon. Like how we used Addu to add, but needed to load the values first before doing so, so the program knows what exactly to add, and where to allocate that memory in the process.