

# Assignment A P1-2 Assembly Program Lab 2. Assign the values into the Array

GitHub Link to your file directly. (Please make it a link, not a text)

<https://github.com/AVC-CS/p1a2-mtvonbargen/blob/0fbf5b58b7af34afefe81d8b46919a6f6999df84/P1A2.asm>

Elaborate on your program

```
##### Data segment #####
.data
    num:    .space 12    # allocate 12 bytes of space for num
    total:  .word 100
##### Code segment #####
.text
.globl main
main:
    la $t0, num          # load (memory) address of 'num' into $t0
    lw $t1, total        # load the value of 'total' into $t1
    sw $t1, 0($t0)       # store the value of $t1 (100) at offset 0 of 'num'
    addi $t1, $t1, 1     # add 1 to the value in $t1 (100), and save it in $t1 (new = 101)
    sw $t1, 4($t0)       # store the value of $t1 (101) at offset 4 of 'num'
    addi $t1, $t1, 1     # add 1 to the value in $t1 (101), and save it in $t1 (new = 102)
    sw $t1, 8($t0)       # store the value of $t1 (102) at offset 8 of 'num'

li $v0, 10 # Exit program
syscall
```

## Program Result

The screenshot displays the QtSpim MIPS simulator interface. The **Regs** window on the left shows the state of the registers, with the **PC** (Program Counter) at 00400048. The **Text Segment** window in the center shows the assembly code being executed, with the **Execution speed** slider set to 100%. The **Data Segment** window on the right shows the memory layout, including the **User Data Segment** and **Kernel Data Segment**. The **Output** window at the bottom shows the execution progress, including the **Log** of the program's execution.

**Regs** (Hex/Dec/Bin):

- Special Registers: PC = 00400048, EPC = 00000000, Cause = 00000000, BadAddr = 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) = 7ffffff8, R6 (a2) = 7ffffffc, R7 (a3) = 00000000, R8 (t0) = 10010000, R9 (t1) = 00000000, R10 (t2) = 00000000, 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.

**Text Segment** (Kernel text, Instruction value, Source code):

```
[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 $0, 4097 [num] ; 14: la $t0, num # load (memory) address of 'num'
[00400028] lui $1, 4097 ; 15: lw $t1, total # load the value of 'total' into
[0040002c] lw $9, 12($1) ; 16: sw $t1, 0($t0) # store the value of $t1 (100) at
[00400030] sw $9, 0($0) ; 17: addi $t1, $t1, 1 # add 1 to the value in $t1 (100)
[00400034] addi $9, $9, 1 ; 18: sw $t1, 4($t0) # store the value of $t1 (101) at
[00400038] sw $9, 4($0) ; 19: addi $t1, $t1, 1 # add 1 to the value in $t1 (101)
[0040003c] addi $9, $9, 1 ; 20: sw $t1, 8($t0) # store the value of $t1 (102) at
[00400040] sw $9, 8($0) ; 22: li $v0, 10 # Exit program
[00400044] ori $2, $0, 10 ; 23: syscall
[00400048] syscall ; 23: syscall
```

**Data Segment** (Kernel data, Hex, Dec):

**User Data Segment**

```
[10010000] 00000064 00000065 00000066 00000064 d...e...f...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 0] - [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
```

**User Stack** (Hex, Dec):

```
[7ffffff8] 00000000 00000000 7ffffffef .....
[7ffffff0] 7ffffffb1 7ffffffda 7ffffffc0 .....
[7ffffff0] 7ffffffb3 7ffffffa2 00000000 00000000 .....
[7ffffff0] 3d5f0000 68742f2e 702e7369 72676f72 ..../this.progr
[7ffffff0] 4c006d61 3d474e41 54552e43 00382d46 am-LANG=C.UTF-8-
[7ffffff0] 45404f48 6f682f3d 772f656d 755f6265 HOME=/home/web_u
[7ffffff0] 00726573 3d445750 4150002f 2f3d4854 ser-PWD=-/PATH=-
[7ffffff0] 45535500 65773d52 73755f62 4c007265 -USER-web_user-L
[7ffffff0] 414e474f 773d454d 755f6265 00726573 OGNNAME=web_user-
```

R8 (t0) = 10010000	R8 (t0) = 10010000		R8 (t0) = 10010000
R9 (t1) = 00000064	R9 (t1) = 00000065		R9 (t1) = 00000066
R10 (t2) = 00000000	R10 (t2) = 00000000		

#### User Data Segment

[10010000] 00000064 00000065 00000066 00000064 d...e...f

#### Kernel Data Segment

Start of program (\$t1) value is 100, which is 00000064

After one increment value is 101 00000065

End of program value is 102, which is 00000066

Offset can be seen in play under User Data Segment.

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

\$t0 = contains the memory address 0x10010000 of 'num'

0x10010000 - 1st stored \$t1 value location

0x10010004 - 2nd stored and incremented \$t1 value location

0x10010008 - 3rd stored and incremented \$t1 value location

#### The interaction between register and memory space

Memory Space - num: is used to set the array size, which is 12 bytes

Memory Space - total: is used to store the value '100' (our starting value to increment)

\$t0 is used to store the memory address of num

\$t1 is used to store the value of total after every incrementation

#### Program logic

Store an initial value 'total' into the first position of a memory address/block and incrementally store sequentially increasing values in subsequent positions.