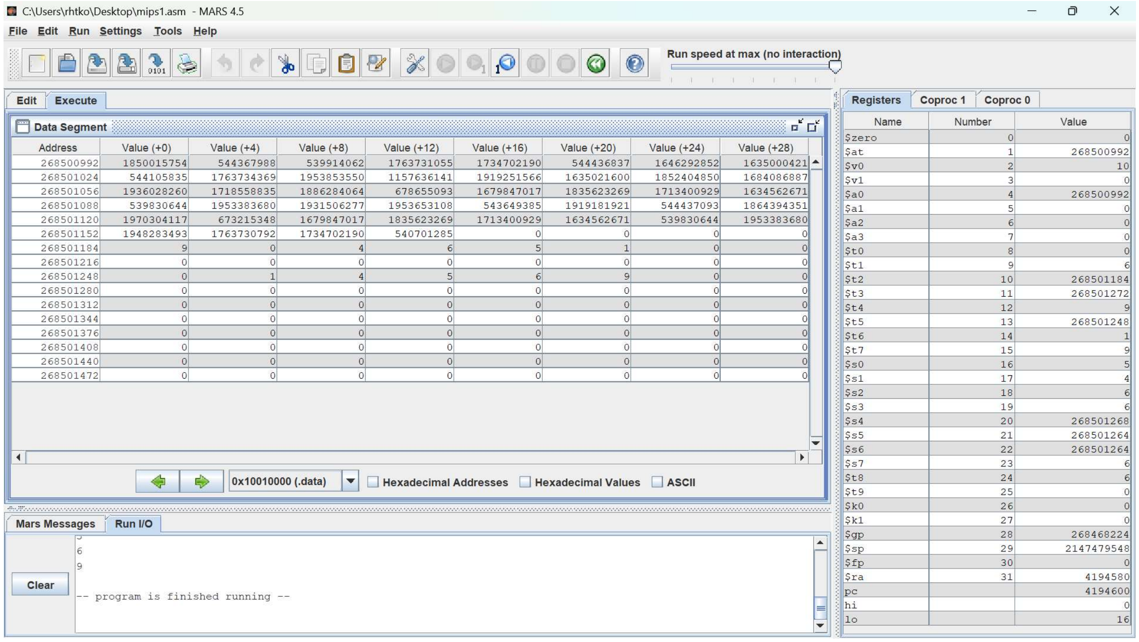
Computer Architecture Assignment-1

1.) Name – Mupparapu Koushik
Roll No. – IMT2022570
Email-ID – Koushik.Mupparapu@iiitb.ac.in
2.) Name – Komaragiri Sai Vishwanath Rohit
Roll No. – IMT2022576
Email-ID – Komaragiri.Sai@iiitb.ac.in

We chose to implement a sorting algorithm in MIPS assembly for the first question. The algorithm we had written is Selection Sort. This code sorts integers in ascending order. When integrated with template.asm, this code first takes input an integer n – number of elements to be stored in the array, then takes input the starting address of the inputs , then the starting address of outputs and finally inputs n integers- the numbers which are to be sorted. For example, if the number of integers to be entered is 6, the starting input address is 268501184, the starting output address is 268501248 and the numbers are entered in the following order 9 0 4 6 5 1,the output will be :

0
1
4
5
6
9

As shown in the Data Segment attached below.


C:\Users\rhtko\Desktop\mips1.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit
Execute
Data Segment
0x10010000 (.data)
Mars Messages
Run I/O
Clear
Registers
Coproc 1
Coproc 0

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1850015754	544367988	539914062	1763731055	1734702190	544436837	1646292852	1635000421
268501024	544105835	1763734369	1953853550	1157636141	1919251566	1635021600	1852404850	1684086887
268501056	1936028260	1718558835	1886284064	678655093	1679847017	1835623269	1713400929	1634562671
268501088	539830644	1953383680	1931506277	1953653108	543649385	1919181921	544437093	1864394351
268501120	1970304117	673215348	1679847017	1835623269	1713400929	1634562671	539830644	1953383680
268501152	1948283493	1763730792	1734702190	540701285	0	0	0	0
268501184	9	0	4	6	5	1	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	1	4	5	6	9	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

□ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

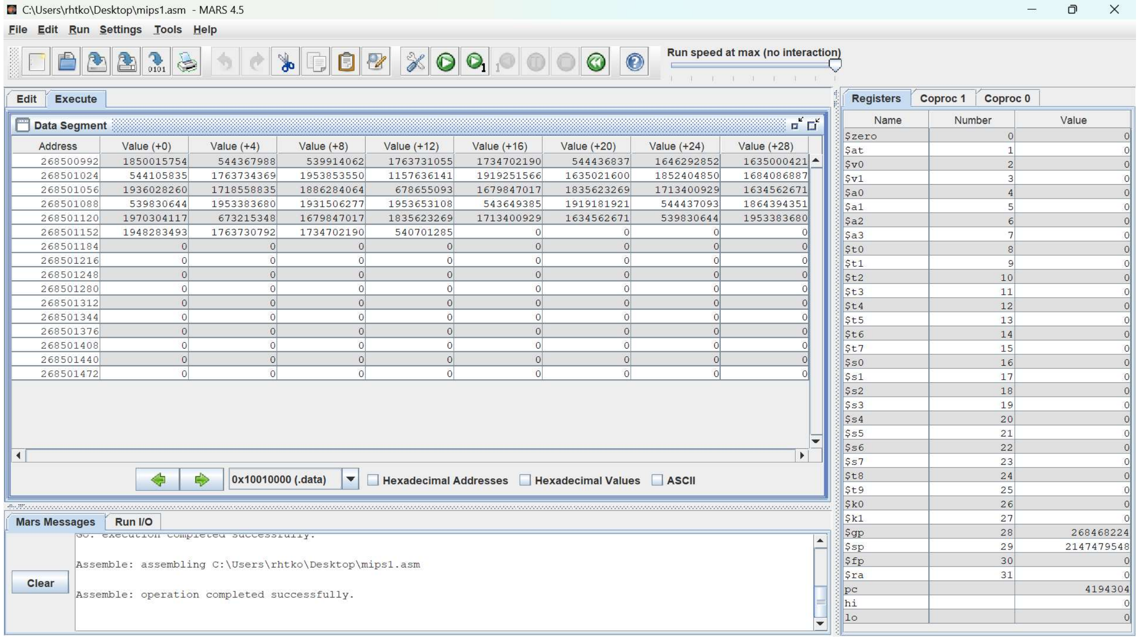
```
6
9

-- program is finished running --
```

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	268500992
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	6
$t2	10	268501184
$t3	11	268501272
$t4	12	9
$t5	13	268501248
$t6	14	1
$t7	15	9
$s0	16	5
$s1	17	4
$s2	18	6
$s3	19	6
$s4	20	268501268
$s5	21	268501264
$s6	22	268501264
$s7	23	6
$t8	24	6
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194580
pc		4194600
hi		0
lo		16

The below table shows the Console taking user input and giving desired output.

```
Enter No. of integers to be taken as input- 6
Enter starting address of inputs(in decimal format)- 268501184
Enter starting address of outputs (in decimal format)- 268501248
Enter the integer: 9
Enter the integer: 0
Enter the integer: 4
Enter the integer: 6
Enter the integer: 5
Enter the integer: 1
0
1
4
5
6
9

-- program is finished running --
```

Values in the registers before the program is executed.

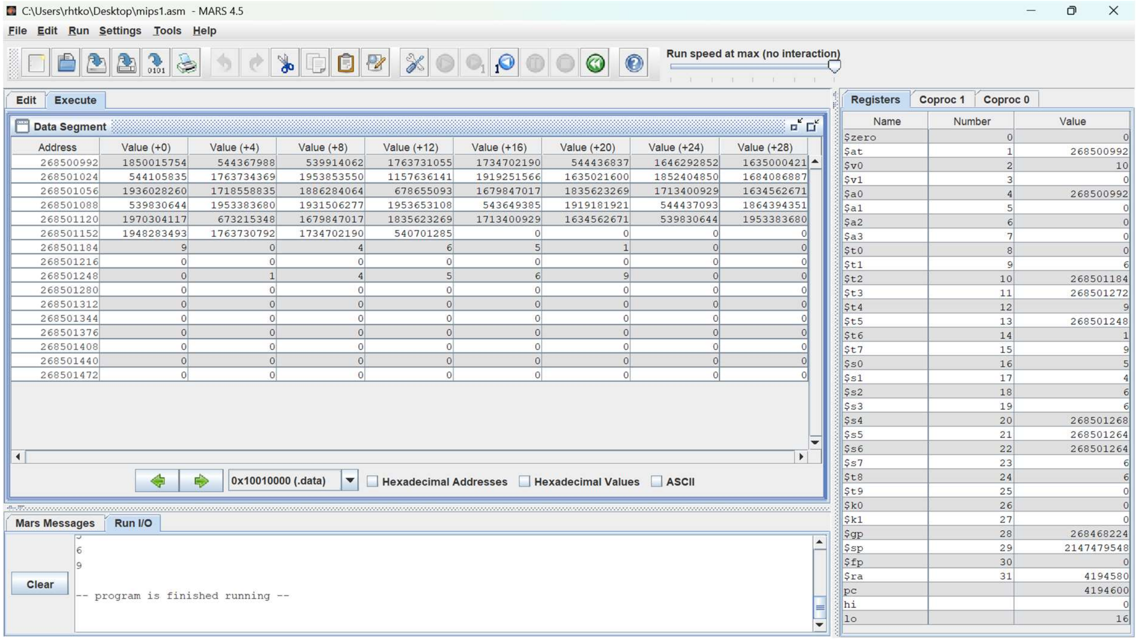


Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1850015754	544367988	539914062	1763731055	1734702190	544436837	1646292852	1635000421
268501024	544105835	1763734369	1953853550	1157636141	1919251566	1635021600	1852404850	1684086887
268501056	1936028260	1718558835	1886284064	678655093	1679847017	1835623269	1713400929	1634562671
268501088	539830644	1953383680	1931506277	1953653108	543649385	1919181921	544437093	1864394351
268501120	1970304117	673215348	1679847017	1835623269	1713400929	1634562671	539830644	1953383680
268501152	1948283493	1763730792	1734702190	540701285	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

```
Go: execution completed successfully.

Assemble: assembling C:\Users\rhtko\Desktop\mips1.asm

Assemble: operation completed successfully.
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

Values in the registers after the program is executed.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1850015754	544367988	539914062	1763731055	1734702190	544436837	1646292852	1635000421
268501024	544105835	1763734369	1953853550	1157636141	1919251566	1635021600	1852404850	1684086887
268501056	1936028260	1718558835	1886284064	678655093	1679847017	1835623269	1713400929	1634562671
268501088	539830644	1953383680	1931506277	1953653108	543649385	1919181921	544437093	1864394351
268501120	1970304117	673215348	1679847017	1835623269	1713400929	1634562671	539830644	1953383680
268501152	1948283493	1763730792	1734702190	540701285	0	0	0	0
268501184	9	0	4	6	5	1	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	1	4	5	6	9	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

0x10010000 (.data) □ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

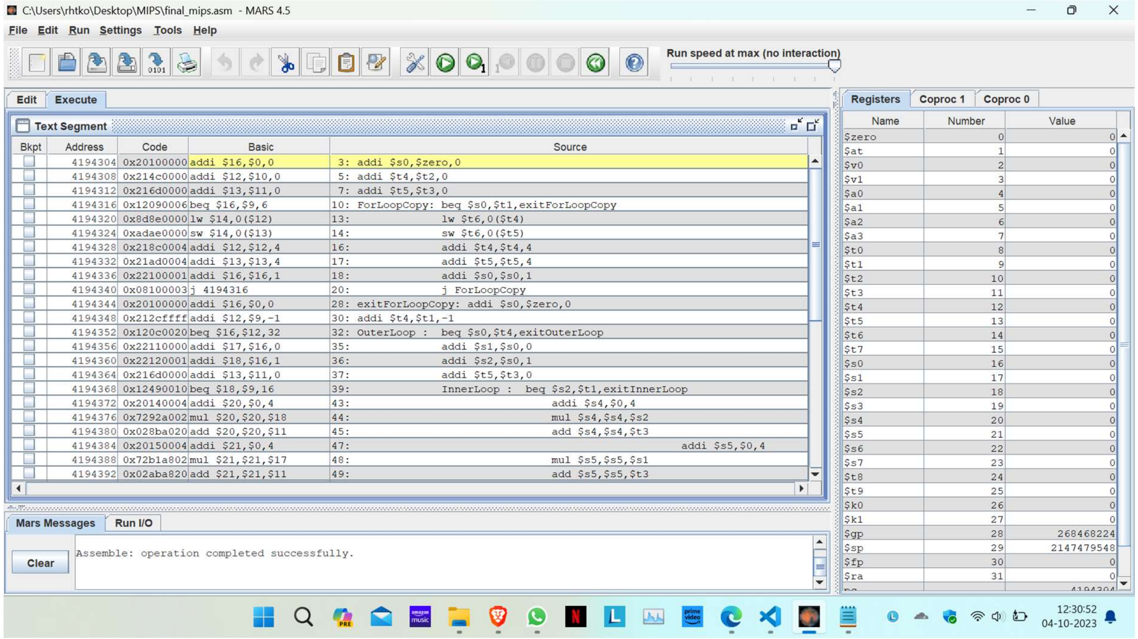
```
6
9

-- program is finished running --
```

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	268500992
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	6
$t2	10	268501184
$t3	11	268501272
$t4	12	9
$t5	13	268501248
$t6	14	1
$t7	15	9
$s0	16	5
$s1	17	4
$s2	18	6
$s3	19	6
$s4	20	268501268
$s5	21	268501264
$s6	22	268501264
$s7	23	6
$t8	24	6
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194580
pc		4194600
hi		0
lo		16

For the second question, we chose to write an assembler in python. In the code, first we read the input file and store it. Then we loop through the file first and store all the labels, instructions and data in the corresponding list or dictionary (data_memory for data, instruction_memory for instructions and labels for labels). Then we run through the instruction_memory and we execute it as if it was being assembled in an assembler and give output in machine code(in hexadecimal).

In the code, we have opcode and registers mapping . The data_memory list stores data values, instruction_memory list stores list of all instructions and labels dictionary stores key-value pairs of all labels in the form of label-address mapping. load_program function processes the code line-by-line and then stores the instructions and data in the corresponding lists. It also identifies the labels and maps them according to their addresses. The assemble_program function translates the assembly code to machine code referring to the corresponding opcode, registers and labels mappings and their addresses.

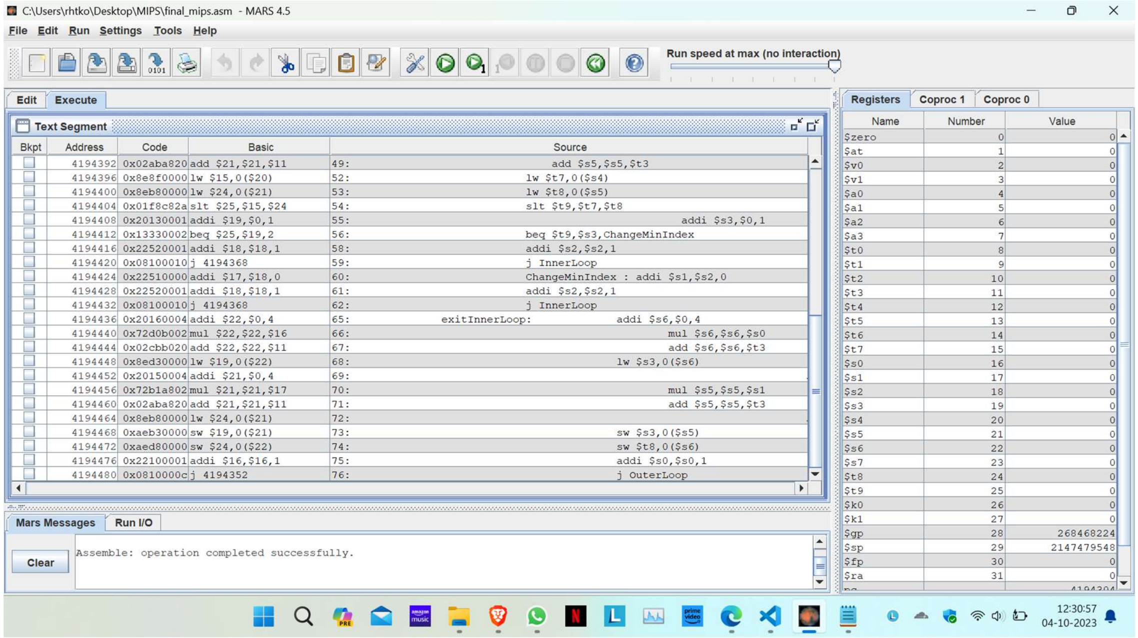
The output(Machine Code) as shown in MARS.


C:\Users\rhtko\Desktop\MIPS\final_mips.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit
Execute
Text Segment
Registers
Coproc 1
Coproc 0
Mars Messages
Run I/O
Clear
12:30:52
04-10-2023

Bkpt	Address	Code	Basic	Source
□	4194304	0x20100000	addi $16,$0,0	3: addi $s0,$zero,0
□	4194308	0x214c0000	addi $12,$10,0	5: addi $t4,$t2,0
□	4194312	0x216d0000	addi $13,$11,0	7: addi $t5,$t3,0
□	4194316	0x12090006	beq $16,$9,6	10: ForLoopCopy: beq $s0,$t1,exitForLoopCopy
□	4194320	0x8d8e0000	lw $14,0($12)	13: lw $t6,0($t4)
□	4194324	0xadae0000	sw $14,0($13)	14: sw $t6,0($t5)
□	4194328	0x218c0004	addi $12,$12,4	16: addi $t4,$t4,4
□	4194332	0x21ad0004	addi $13,$13,4	17: addi $t5,$t5,4
□	4194336	0x22100001	addi $16,$16,1	18: addi $s0,$s0,1
□	4194340	0x08100003	j 4194316	20: j ForLoopCopy
□	4194344	0x20100000	addi $16,$0,0	28: exitForLoopCopy: addi $s0,$zero,0
□	4194348	0x212cffff	addi $12,$9,-1	30: addi $t4,$t1,-1
□	4194352	0x120c0020	beq $16,$12,32	32: OuterLoop : beq $s0,$t4,exitOuterLoop
□	4194356	0x22110000	addi $17,$16,0	35: addi $s1,$s0,0
□	4194360	0x22120001	addi $18,$16,1	36: addi $s2,$s0,1
□	4194364	0x216d0000	addi $13,$11,0	37: addi $t5,$t3,0
□	4194368	0x12490010	beq $18,$9,16	39: InnerLoop : beq $s2,$t1,exitInnerLoop
□	4194372	0x20140004	addi $20,$0,4	43: addi $s4,$0,4
□	4194376	0x7292a002	mul $20,$20,$18	44: mul $s4,$s4,$s2
□	4194380	0x028ba020	add $20,$20,$11	45: add $s4,$s4,$t3
□	4194384	0x20150004	addi $21,$0,4	47: addi $s5,$0,4
□	4194388	0x72b1a802	mul $21,$21,$17	48: mul $s5,$s5,$s1
□	4194392	0x02aba820	add $21,$21,$11	49: add $s5,$s5,$t3

```
Assemble: operation completed successfully.
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304

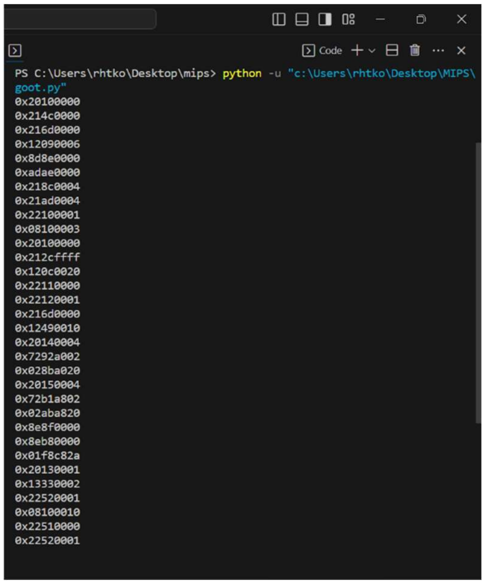

C:\Users\rhtko\Desktop\MIPS\final_mips.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Mars Messages Run I/O
Clear
Registers Coproc 1 Coproc 0
12:30:57
04-10-2023

Bkpt	Address	Code	Basic	Source
□	4194392	0x02aba820	add $21,$21,$11	49: add $s5,$s5,$t3
□	4194396	0x8e8f0000	lw $15,0($20)	52: lw $t7,0($s4)
□	4194400	0x8eb80000	lw $24,0($21)	53: lw $t8,0($s5)
□	4194404	0x01f8c82a	slt $25,$15,$24	54: slt $t9,$t7,$t8
□	4194408	0x20130001	addi $19,$0,1	55: addi $s3,$0,1
□	4194412	0x13330002	beq $25,$19,2	56: beq $t9,$s3,ChangeMinIndex
□	4194416	0x22520001	addi $18,$18,1	58: addi $s2,$s2,1
□	4194420	0x08100010	j 4194368	59: j InnerLoop
□	4194424	0x22510000	addi $17,$18,0	60: ChangeMinIndex : addi $s1,$s2,0
□	4194428	0x22520001	addi $18,$18,1	61: addi $s2,$s2,1
□	4194432	0x08100010	j 4194368	62: j InnerLoop
□	4194436	0x20160004	addi $22,$0,4	65: exitInnerLoop: addi $s6,$0,4
□	4194440	0x72d0b002	mul $22,$22,$16	66: mul $s6,$s6,$s0
□	4194444	0x02cbb020	add $22,$22,$11	67: add $s6,$s6,$t3
□	4194448	0x8ed30000	lw $19,0($22)	68: lw $s3,0($s6)
□	4194452	0x20150004	addi $21,$0,4	69:
□	4194456	0x72b1a802	mul $21,$21,$17	70: mul $s5,$s5,$s1
□	4194460	0x02aba820	add $21,$21,$11	71: add $s5,$s5,$t3
□	4194464	0x8eb80000	lw $24,0($21)	72:
□	4194468	0xaeb30000	sw $19,0($21)	73: sw $s3,0($s5)
□	4194472	0xaed80000	sw $24,0($22)	74: sw $t8,0($s6)
□	4194476	0x22100001	addi $16,$16,1	75: addi $s0,$s0,1
□	4194480	0x0810000c	j 4194352	76: j OuterLoop

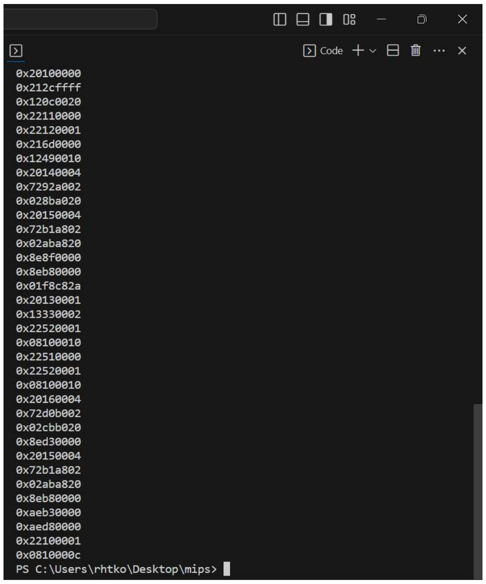
```
Assemble: operation completed successfully.
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304

The output(Machine Code) as shown in Python by the MIPSAssembler in Python.



```
PS C:\Users\rhtko\Desktop\mips> python -u "c:\Users\rhtko\Desktop\MIPS\
goot.py"
0x20100000
0x214c0000
0x216d0000
0x12090006
0x8d8e0000
0xadae0000
0x218c0004
0x21ad0004
0x22100001
0x08100003
0x20100000
0x212cffff
0x120c0020
0x22110000
0x22120001
0x216d0000
0x12490010
0x20140004
0x7292a002
0x028ba020
0x20150004
0x72b1a802
0x02aba820
0x8e8f0000
0x8eb80000
0x01f8c82a
0x20130001
0x13330002
0x22520001
0x08100010
0x22510000
0x22520001
```



```
0x20100000
0x212cffff
0x120c0020
0x22110000
0x22120001
0x216d0000
0x12490010
0x20140004
0x7292a002
0x028ba020
0x20150004
0x72b1a802
0x02aba820
0x8e8f0000
0x8eb80000
0x01f8c82a
0x20130001
0x13330002
0x22520001
0x08100010
0x22510000
0x22520001
0x08100010
0x20160004
0x72d0b002
0x02cbb020
0x8ed30000
0x20150004
0x72b1a802
0x02aba820
0x8eb80000
0xaeb30000
0xaed80000
0x22100001
0x0810000c
PS C:\Users\rhtko\Desktop\mips>
```