

Computer Organization 2017

HOMEWORK I

Overview

The goal of this homework is to help you understand **the MIPS instruction set**. In this homework, you need to write **a assemble program** which can **implement Insertion Sort** by using MIPS instructions. In addition, you need to verify your assemble program by using Qtspim. Please follow the instruction table in this homework and satisfy all the homework requirements. **DO NOT** using MIPS instructions without listed in the table. You also need to take a snapshot and explain your assemble program in your report.

General rules for deliverables

- You need to complete this homework **INDIVIDUALLY**. You can discuss the homework with other students, but you need to do the homework by yourself. If you copy your codes from someone else, you **will not get any scores**.
- When submitting your homework, compress all files into a single **zip** file, and upload the compressed file to Moodle.
 - Please follow the file hierarchy shown in Figure 1.

F740XXXXX(your id)(folder)

SRC(folder) * Store your source code

F740XXXXX_Report.docx (Project Report. The report template is already included.

Follow the template to complete the report.)

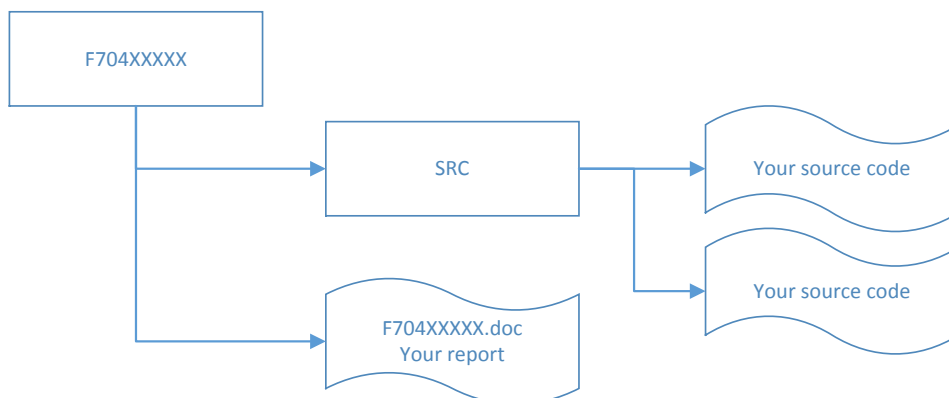


Figure 1. File hierarchy for homework submission

- **Important! DO NOT submit your homework in the last minute. Late submission is not accepted.**
- You should finish **all the requirements (shown below) in this homework** and Project report.

Homework Description

Insertion sort is a simple sorting algorithm that very easy to implement. Figure 2 shows the steps for sorting the sequence {3, 7, 4, 9, 5, 2, 6, 1}. In each step, the element under consideration is underlined. The element that was moved (or left in place because it was biggest yet considered) in the previous step is shown in red. Figure 3 shows the insertion sort pseudo code.

Iteration	Result
0	<u>3</u> 7 4 9 5 2 6 1
1	3 <u>7</u> 4 9 5 2 6 1
2	3 7 <u>4</u> 9 5 2 6 1
3	3 4 7 <u>9</u> 5 2 6 1
4	3 4 7 9 <u>5</u> 2 6 1
5	3 4 5 7 9 <u>2</u> 6 1
6	2 3 4 5 7 9 <u>6</u> 1
7	2 3 4 5 6 7 9 <u>1</u>
8	1 2 3 4 5 6 7 9

Figure 2. Steps of Insertion sort

```

for i ← 1 to length(A)
  j ← i
  while j > 0 and A[j-1] > A[j]
    swap A[j] and A[j-1]
  j ← j - 1
end while
end for

```

Figure 3. Pseudo code of Insertion sort

Assembler Syntax

instruction	rd	rs	rt
-------------	----	----	----

R-type Instruction Machine Code Format

opcode	rs	rt	rd	shamt	funct
31	26 25	21 20	16 15	11 10	6 5 0

opcode	Mnemonics	SRC1	SRC2	DST	funct	Description
000000	nop	00000	00000	00000	000000	No operation
000000	add	\$Rs	\$Rt	\$Rd	100000	Rd = Rs + Rt
000000	sub	\$Rs	\$Rt	\$Rd	100010	Rd = Rs - Rt
000000	and	\$Rs	\$Rt	\$Rd	100100	Rd = Rs & Rt
000000	or	\$Rs	\$Rt	\$Rd	100101	Rd = Rs Rt
000000	xor	\$Rs	\$Rt	\$Rd	100110	Rd = Rs ^ Rt
000000	nor	\$Rs	\$Rt	\$Rd	100111	Rd = ~(Rs Rt)
000000	slt	\$Rs	\$Rt	\$Rd	101010	Rd = (Rs < Rt)?1:0
000000	sll		\$Rt	\$Rd	000000	Rd = Rt << shamt
000000	srl		\$Rt	\$Rd	000010	Rd = Rt >> shamt
000000	jr	\$Rs			001000	PC=Rs
000000	jalr	\$Rs			001001	R[31] = PC + 8 ; PC=Rs

Figure 4. R-type MIPS instructions

Assembler Syntax

instruction	rt	rs	imm
-------------	----	----	-----

I-type Instruction Machine Code Format

opcode	rs	rt	immediate
31	26 25	21 20	16 15 0

opcode	Mnemonics	SRC1	DST	SRC2	Description
001000	addi	\$Rs	\$Rt	imm	$Rt = Rs + imm$
001100	andi	\$Rs	\$Rt	imm	$Rt = Rs \& imm$
001010	slti	\$Rs	\$Rt	imm	$Rt = (Rs < imm) ? 1 : 0$
000100	beq	\$Rs	\$Rt	imm	If($Rs == Rt$) $PC = PC + 4 + imm$
000101	bne	\$Rs	\$Rt	imm	If($Rs != Rt$) $PC = PC + 4 + imm$
100011	lw	\$Rs	\$Rt	imm	$Rt = Mem[Rs + imm]$
100001	lh	\$Rs	\$Rt	imm	$data = Mem[Rs + imm]$ $Rt = data[15:0] \leftarrow \text{Sign-extend } 16\text{bits}$
101011	sw	\$Rs	\$Rt	imm	$Mem[Rs + imm] = Rt$
101001	sh	\$Rs	\$Rt	imm	$data \leftarrow Rt[15:0] \text{ Sign-extend } 16\text{bits}$ $Mem[Rs + imm] = Rt$

Figure 5. I-type MIPS instructions

Assembler Syntax

instruction	Target(label)
-------------	---------------

J-type Instruction Machine Code Format

opcode	address
31	26 25 0

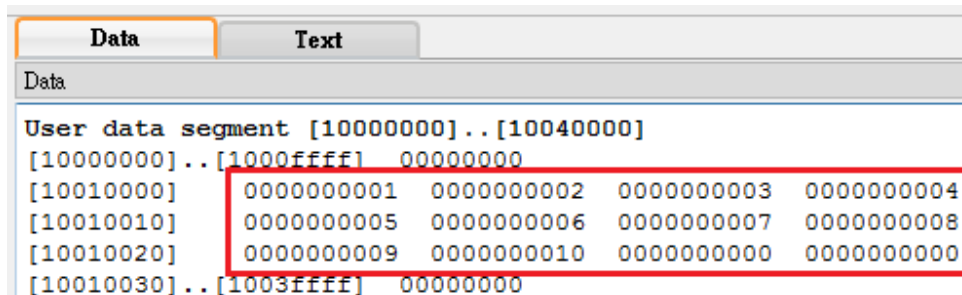
opcode	Mnemonics	Address	Description
000010	j	jumpAddr	$PC = \text{jumpAddr}$
000011	jal	jumpAddr	$R[31] = PC + 8 ; PC = \text{jumpAddr}$

Figure 6. J-type MIPS instructions

Homework Requirements

1. Implement Insertion sort according to the MIPS instruction table given above.
2. Using MIPS Simulator (Qtspim) to run your assembly code to sort the sequence of size ten and **store result into user data segment**.
3. Finish your Project report

Note: please take snapshot of your result and paste into your report (e.g. Figure 7).



The screenshot shows the 'Data' window of a MIPS simulator. It displays the 'User data segment' from address [10000000] to [10040000]. A table of memory contents is shown, with a red box highlighting the first ten elements of the array, which are sorted in ascending order. The values are 000000001 through 000000010.

Address	Value
[10000000]..[1000ffff]	00000000
[10010000]	0000000001
[10010004]	0000000002
[10010008]	0000000003
[10010010]	0000000005
[10010014]	0000000006
[10010018]	0000000007
[10010020]	0000000009
[10010024]	0000000010
[10010030]..[1003ffff]	00000000

Figure 7. Snapshot of correct Result

Important

When you upload your file, please check you have done and followed all requirements, including **File hierarchy**, **Requirement file** and **Report format**.

If you have any questions, please contact us.