ECS 409 : MIPS Assembly Programming
Sequential Construct-II Assignment
Floating Point Instructions
Input Output Instructions
Comparison Instructions

Instructor: Dr. Sukarn Agarwal,
Assistant Professor,
EECS,
IISER Bhopal

October 14, 2025

Sequential Construct

The programming requires a dividing a task, into small unit of work. These unit of work are represented with programming construct that represents part of task. In the sequential construct, the designated task is broken into smaller task one follow by another.

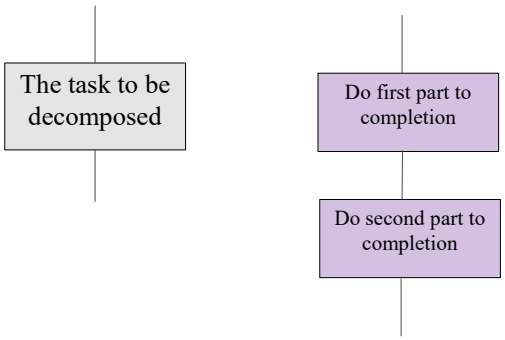

Figure 1: Representational view of Sequential Construct

Floating Point Instructions

MIPS provide several set of floating point instructions:

- Arithmetic
- Data Movement
- Conditional Jump

In this assignment, we have discussed only Arithmetic and Data Movement Instructions. Note that the details and list of the register is already provided in the instruction manual.

Instruction	Example	Meaning	Comments
Arithmetic Instructions			
add	add.s $f0, $f1, $f2	$f0=$f1+$f2	none
subtract	sub.s $f0, $f1, $f2	$f0=$f1-$f2	none
multiply	mul.s $f0, $f1, $f2	$f0=$f1*$f2	none
division	div.s $f0, $f1, $f2	$f0=$f1/$f2	none
absolute	abs.s $f0, $f1	$f0=—$f1—	Absolute Value of floating point number
negative	neg.s $f0, $f1	$f0=-$f1	Negate the floating point number
Data Movement and Conversion Instructions			
load float	l.s $f0, 100($t2)	$f0=Mem[$t2+100]	Copy from Memory to Floating Point Register
store float	s.s $f0, 100($t2)	Mem[$t2+100]=$f0	Copy from Floating Point Register to Memory
load float immediate	li.s $f0, 10.0	$f0=10.0	Load Floating point immediate value into Register
move	move.s $f0, $f1	$f0=$f1	Copy from register to register
convert to integer	cvt.w.s $f2, $f4	$f2=$f4	Convert from single precision FP (f4) to integer (f2)
convert to float	cvt.s.w $f2, $f4	$f2=$f4	Convert from integer (f2) to single precision (f4)

Table 1: Floating Point Instructions with their details and explanations

System Call Related to I/O

System calls are used for input, output and to exit the program. These calls are commences with the help of *syscall* function. To use the instruction, the appropriate arguments in registers $v0, $a0-$a1, or $f12 are supplied depending on the specific call required. The system call will return the result values into the register based on the datatype and operation conducted.

Following are the set of system call that is to be used in this assignment.

Service	Operation	Code (in $v0)	Arguement	Results
print_int	Print integer number (32 bit)	1	$a0=integer to be printed	none
print_float	Print floating-point number (32 bit)	2	$f12=float to be printed	none
print_double	Print floating-point number (64 bit)	3	$f12=float to be printed	none
print_string	Print null-terminated character string	4	$a0=address of string in memory	none
read_int	Read integer number from user	5	none	integer written in $v0
read_float	Read floating-point number from user	6	none	float written in $f0
read_double	Read double floating point number from user	7	none	double written in $f0
read_string	Work the same as standard C library fgets()	8	$a0=memory address of string input buffer $a1=length of string buffer (n)	none
sbrk	Returns the address to a block of memory containing n additional bytes (dynamic memory allocation)	9	$a0=amount	address in $v0
print_char	Print Character	11	$a0=character to be printed	none
read_char	Read Character from user	12	none	char written in $v0

Table 2: List of System Calls with their usage and explanations

Comparison Instructions

Following are the set of comparison instructions that is to be used in this assignment.

Instruction	Example	Meaning	Comments
set on less than	slt $t1, $t2, $t3	if($t2<$t3)$t1=1; else $t1=0	Test if less than. If true, set $t1 to 1. Otherwise set $t1 to 0
set on less than immediate	slti $t1, $t2, 100	if($t2<100)$t1=1; else $t1=0	Test if less than. If true, set $t1 to 1. Otherwise set $t1 to 0
set equal	seq $t1, $t2, $t3	if($t2=$t3)$t1=1; else $t1=0	Test if equal. If true, set $t1 to 1. Otherwise set $t1 to 0
set greater than equal	sge $t1, $t2, $t3	if($t2>=$t3)$t1=1; else $t1=0	Test if greater than equal. If true, set $t1 to 1. Otherwise set $t1 to 0.
set greater than	sgt $t1, $t2, $t3	if($t2>$t3)$t1=1; else $t1=0	Test if greater than. If true, set $t1 to 1. Otherwise set $t1 to 0.
set less than equal	sle $t1, $t2, $t3	if($t2<=$t3)$t1=1; else $t1=0	Test if less than equal. If true, set $t1 to 1. Otherwise set $t1 to 0.
set not equal	sne $t1, $t2, $t3	if($t2! =$t3)$t1=1; else $t1=0	Test if not equal. If true, set $t1 to 1. Otherwise set $t1 to 0.

Table 3: List of Comparison Instructions with their details and explanations

Problem 1: Write an MIPS assembly program that takes user first name as an input and prompt the message

```
Hi Ajay, MIPS assembly programming is very exciting to learn
```

Problem 2: Write an assembly program that takes principle amount, rate of interest and time (taken as an input to integer register) as an input from the user and calculate simple interest and display the raw and absolute results to the user.

Problem 3: Write an assembly program that takes three character string from the user and print the second character from it. (**Hint:** use lbu instruction).

Problem 4: Write an assembly program that takes two strings (of length of two characters) from the user and calculate the hamming distance. For example if "hi" and "he" are the input from the user then the hamming distance is 01.

Note: Usage of other instructions (other than the instructions given in the tables 1,2,3 and the instructions given in assignment-2) to solve the problem results into the zero marks. Submit all of your source code and final screen shot of the register panels (both integer and floating point) to the google classroom portal (in a pdf format, make sure that your name appear on the screen) on the end of the day of 26th Oct 2025 (Indian Standard Time). Further any copy case between the assignments results into the zero marks.