CS312 : MIPS Assembly Programming
Assignment 8: Stacks

Instructor: Dr. Sukarn Agarwal,
Assistant Professor,
Department of CSE,
IIT (BHU) Varanasi

April 7, 2021

Stacks

The stack is a memory area used to save local variables. A certain chunk of main memory is reserved for the stack, called stack space/area in the MIPS machine. The following points need to be considered while working with stack in MIPS:

- The stack expands downwards in terms of memory addresses.
- The stack's top element's memory address is stored in the special purpose register, called Stack Pointer (`sp`).
- MIPS does not provide any `push` and `pop` statement. Instead, it will be explicitly handled by the MIPS assembly programmer.

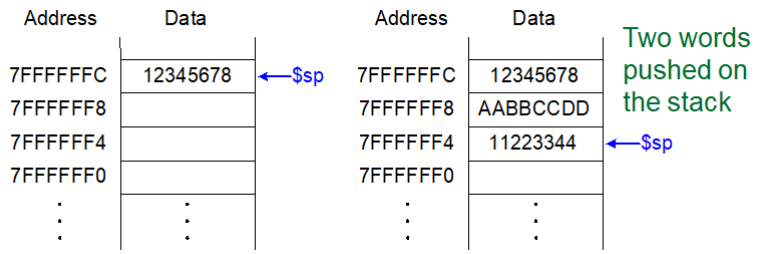

Figure 1: Representational view of Stack

Pushing an Element in the stack In order to place the data or address element in the stack, the following are the two steps that are necessary to follow:

- Progress the stack pointer, `sp` to the down to make space for the newly added element.
- Store the element into the stack.

Following are the two sample example code to push the data elements from the register `$t7` and `$t9` into the stack.

One Way:

```
sub $sp, $sp, 8
sw $t9, 4($sp)
sw $t7, 8($sp)
```

Alternate Way:

```
sw $t9, -4($sp)
sw $t7, -8($sp)
sub $sp, $sp, 8
```

Accessing and Popping Elements With stack pointer (`$sp`), you can access an element in the stack at any position (not just the top one) if and only if you know the position relative to top element `$sp`.

For example, to retrieve the value of `$t7`:

```
lw $a0, 8($sp)
```

With the above command, you can also `pop` or "wipe" the element simply by making the stack pointer upward. For example, to pop the value of `$t9` that was previously added, yielding the stack shown at the bottom:

```
addi $sp, $sp, 8
```

Note that the popped data is still present in memory, but the data past the stack pointer is considered invalid.

Problem 1: Write a MIPS assembly program that let the user to enter 10 number as an input and output the number in the reverse order using stack. The one possible output format is as follows:

```
Enter the 10 input number: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
Reverse String: 0, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

Problem 2: Write a MIPS assembly program that lets the user enter the arithmetic expression with braces and parenthesis. The MIPS program checks whether the given arithmetic expression is correct or not in terms of the number of opening and closing braces. The possible sample output formats for this MIPS assembly program are as follows:

```
1. Enter the airthmetic expression: (a+(b*c))
The airthmetic expression is correct in
terms of number of opening and closing paranthesis.
Number of opening braces: 2
Number of closing braces: 2

2. Enter the airthmetic expression: )a+(b*c)(
The airthmetic expression is incorrect in
 terms of number of opening and closing paranthesis.
Number of opening braces: 2
Number of closing braces: 2
```

Problem 3: Write a MIPS assembly function to perform an absolute value vector addition. Use the stack to pass parameters. The parameters are the starting address of three different word arrays (vectors): X, Y, Z, and an integer values N and M (assume that it is initialized and given with the main function) specifying the size of the vectors (Y and Z). If overflow ever occurs when executing this function, it should return an error status of "1", and the function aborts any further processing. Otherwise, return the value "0" for status. The function will perform the following vector addition:

```
Xi  =  | Yi  |   +  | Zi | ; with  i  going from 0 to N - 1 / M - 1.
```

Also write a main program to test this function. The output format of this function (if there is no overflow, value of vector Y and Z to be 1, 3, 2, 5 and 2, 4, 6, 4 and the values of N and M are set to 4) is as follows:

```
The addition product of vector X: 3, 7, 8, 9
There is no overflow incur in the vector addition
```

The other output format of this function (if there is overflow, value of vector Y and Z to be 1, 3, 2, 5 and 2, 4, 6, 4, 5 and the values of N and M are set to 4 and 5) is as follows:

```
There is overflow incur in the vector addition.
Array Y size is 4.
Array Z size is 5.
```

Problem 4: Write a MIPS assembly function to return the Nth element in the Fibonacci sequence. A value N is passed to the function on the stack, and the Nth Fibonacci number E is returned on the stack. If N is greater than 60, overflow will occur, so return a value of -1 if N is greater than 60. The first few numbers in the Fibonacci sequence are: 0, 1, 1, 2, 3, 5 . . . .One of the possible output formats are (for N values to be 5 and 62) as follows:

```
1.    5 element in fibonaaci series is 3
2.    62 element in fibonacci series cannot be genreated,
 Overflow condition incurs in this case.
```

Note: Submit all of your source code and the final screenshots of the console and register panels (both integer and floating-point) to the google classroom portal by the end of the day of 10 April 2021 (Indian Standard Time). Further, any copy case between the assignments results in zero marks. In case of any doubt(s) regarding the assignment, you can contact TA: Nirbhay and Deepika.