Macros

- Uses:
  - to improve readability of your program
  - to eliminate repetitive code construction
  - to reduce overhead associated with subroutines
- Register usage must be carefully considered
- Register Approaches:
  - New Pseudo Instruction: use the $at register -- but don't use any pseudo instructions
  - Marshalling: use only the registers provided as arguments
  - Inlined- subroutine: Utilize: $a0, $a1, $a2, $a3, $v0, and $v1
- Syntax:

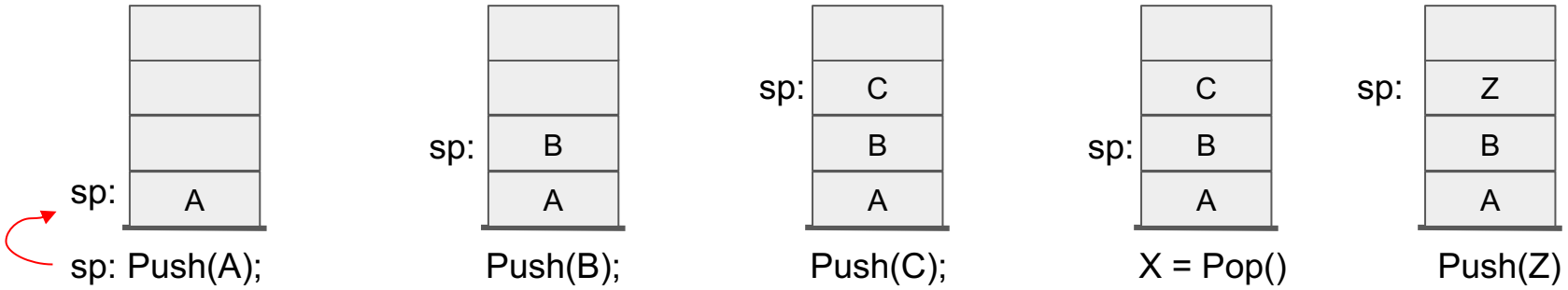
```
.macro  <name>(%arg1 .. %argn)
    <list of native instructions>
.end_macro
```

Stack Operations

```
Push(a) ⇔
    sp = sp + 1
    sp[0] = a
```

```
x = Pop() ⇔
        x = sp[0]
        sp = sp - 1
```

- Stack is an abstract data structure
- The stack is an array of words
- Operations:
  - Push: Push(A), Push(B), Push(C)
  - Pop: X = Pop();
  - Push: Push(Z);



But the MIPS Way

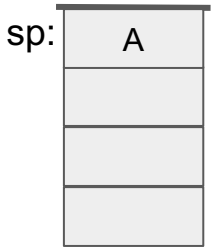
- Stack is an abstract data structure
- Operations:
  - Push: Push(A), Push(B), Push(C)
  - Pop: X = Pop();
- sp: points to the current top of stack

```
Push(a) ⇔
    sp = sp - 1
    sp[0] = a
```

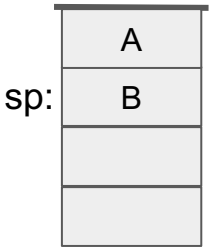
```
x = Pop() ⇔
        x = sp[0]
        sp = sp + 1
```

```
Push(a) ⇔
    subi $sp, $sp, 4
    sw $a0, 0($sp)
```

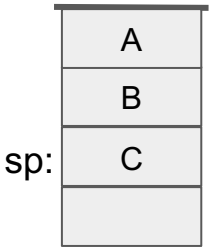
```
x = Pop() ⇔
        lw $v0, 0($sp)
        addi $sp, $sp, 4
```



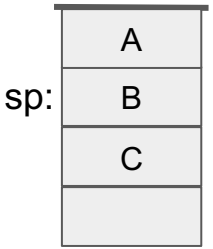
Push(A);



Push(B);



Push(C);

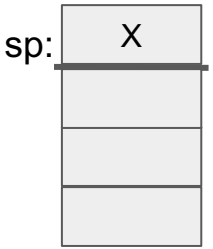


X = Pop()

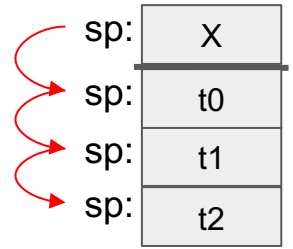
Multiple Pushes / Pops

```
Push(a) ⇔
    sp = sp - 1
    sp[0] = a
```

```
x = Pop() ⇔
        x = sp[0]
        sp = sp + 1
```

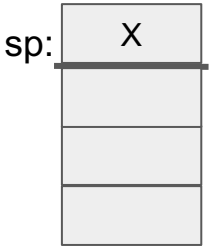



```
Push(t0);
Push(t1);
Push(t2);
```

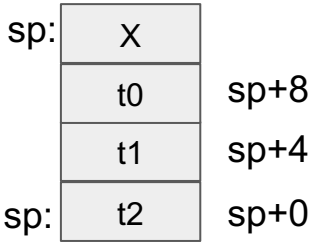



```
Push(a) ⇔
    subi $sp, $sp, 4
    sw $a0, 0($sp)
```

```
x = Pop() ⇔
        lw $v0, 0($sp)
        addi $sp, $sp, 4
```




```
subi $sp, $sp, 12
sw $t0, 8($sp)
sw $t1, 4($sp)
sw $t2, 0($sp)
```




```
t0 = Pop();
t1 = Pop();
t2 = Pop();
```

```
lw $t0, 8($sp)
lw $t1, 4($sp)
lw $t2, 0($sp)
addi $sp, $sp, 12
```

Printf: print(variable); println("string");

- Java Prototype
  - public PrintStream printf(String format, Object... args)
- Java Example
  - printf("the value of x is %d", x);
  - printf("x=%d, y=%d, z=%d\n", x, y, z);
- Format Specifier,
  - %conversion
- Format Conversions:
  - d: decimal, u: unsigned decimal
  - o: octal
  - x: hexadecimal
  - c: character
  - s: string
  - f: floating point
  - ~~t: binary (bi"t")~~

- MIPS Macros:
  - print_d, print_u, print_di
  - print_o
  - print_x
  - print_c
  - print_s
  - ~~print_f~~
  - print_t