숭실대학교
Soongsil University

Computer Architecture

강의 #3: Instructions: Language of the Machine (추가)

2021년 1학기

Young Geun Kim (김영근)

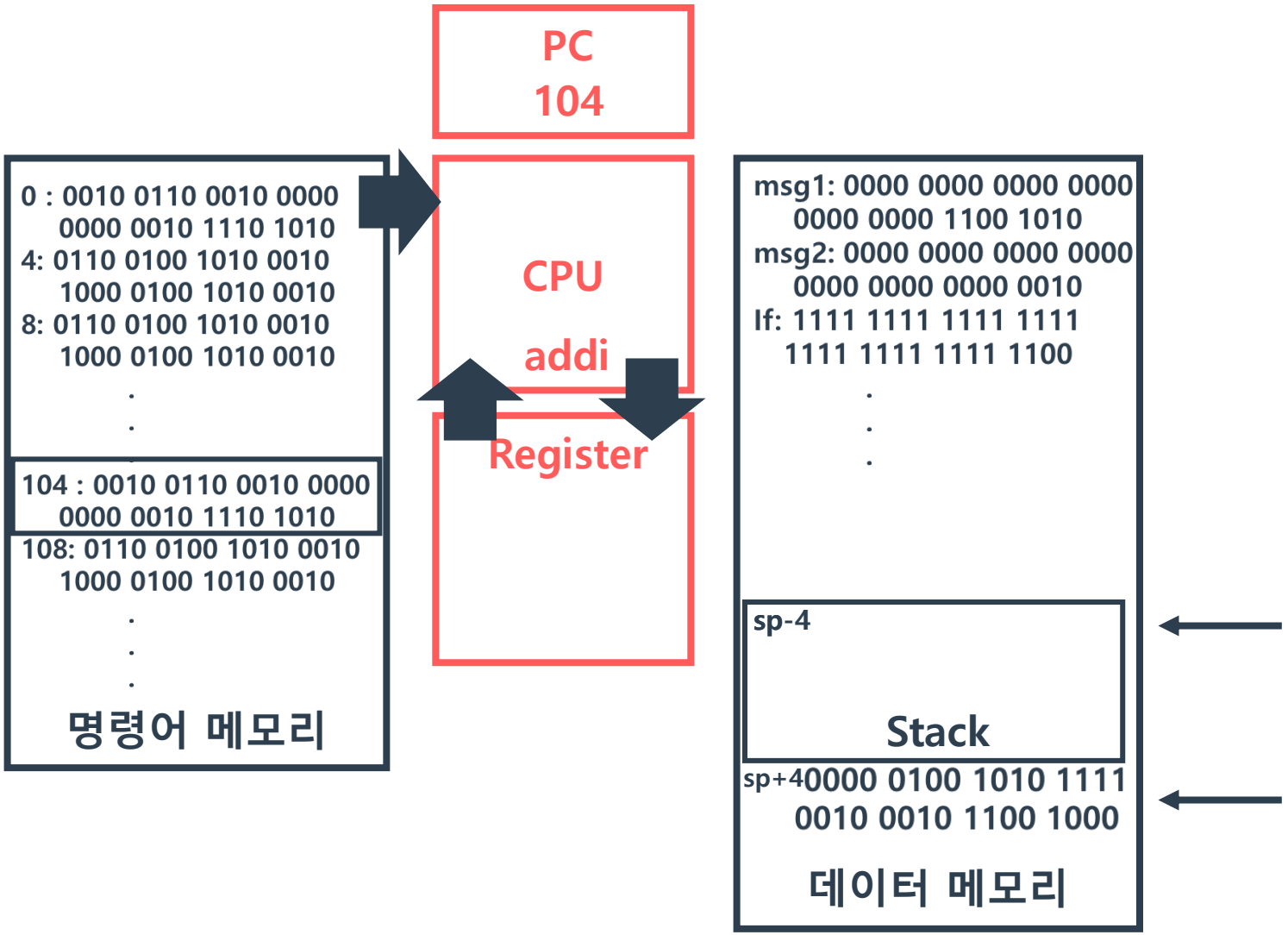
Procedure Call

```
# ------------------------------------------------------------------

        .text

        .globl  main
main:
        # Register assignments
        # $s0 = N
        # $s1 = counter (i)
        # $s2 = sum

        # Print msg1
0       li      $v0,4           # print_string syscall code = 4
4       la      $a0, msg1
8       syscall

        # Get N from user and save
12      li      $v0,5           # read_int syscall code = 5
16      syscall                 = N
20      move    $s0,$v0         # syscall results returned in $v0

        # Initialize registers
24      li      $s1, 0          # Reg $s1 = counter (i)
28      li      $s2, 0          # Reg $s2 = sum

        # Main loop body
32  loop:   addi    $s1, $s1, 1     # i = i + 1

        # Call add function
36      move    $a0, $s2        # Argument 1: sum ($s2)
40      move    $a1, $s1        # Argument 2: i ($s1)
44      jal     add2    = 104   # Save current PC in $ra, and jump to add2
48      move    $s2,$v0         # Return value saved in $v0. This is sum ($s2)
52      beq     $s0, $s1, exit  # if i = N, continue
56      j       loop

        # Exit routine - print msg2
60  exit:   li      $v0, 4          # print_string syscall code = 4
64      la      $a0, msg2
68      syscall

        # Print sum
72      li      $v0,1           # print_string syscall code = 4
76      move    $a0, $s2
80      syscall

        # Print newline
84      li      $v0,4           # print_string syscall code = 4
88      la      $a0, lf
92      syscall
96      li      $v0,10          # exit
100     syscall
```

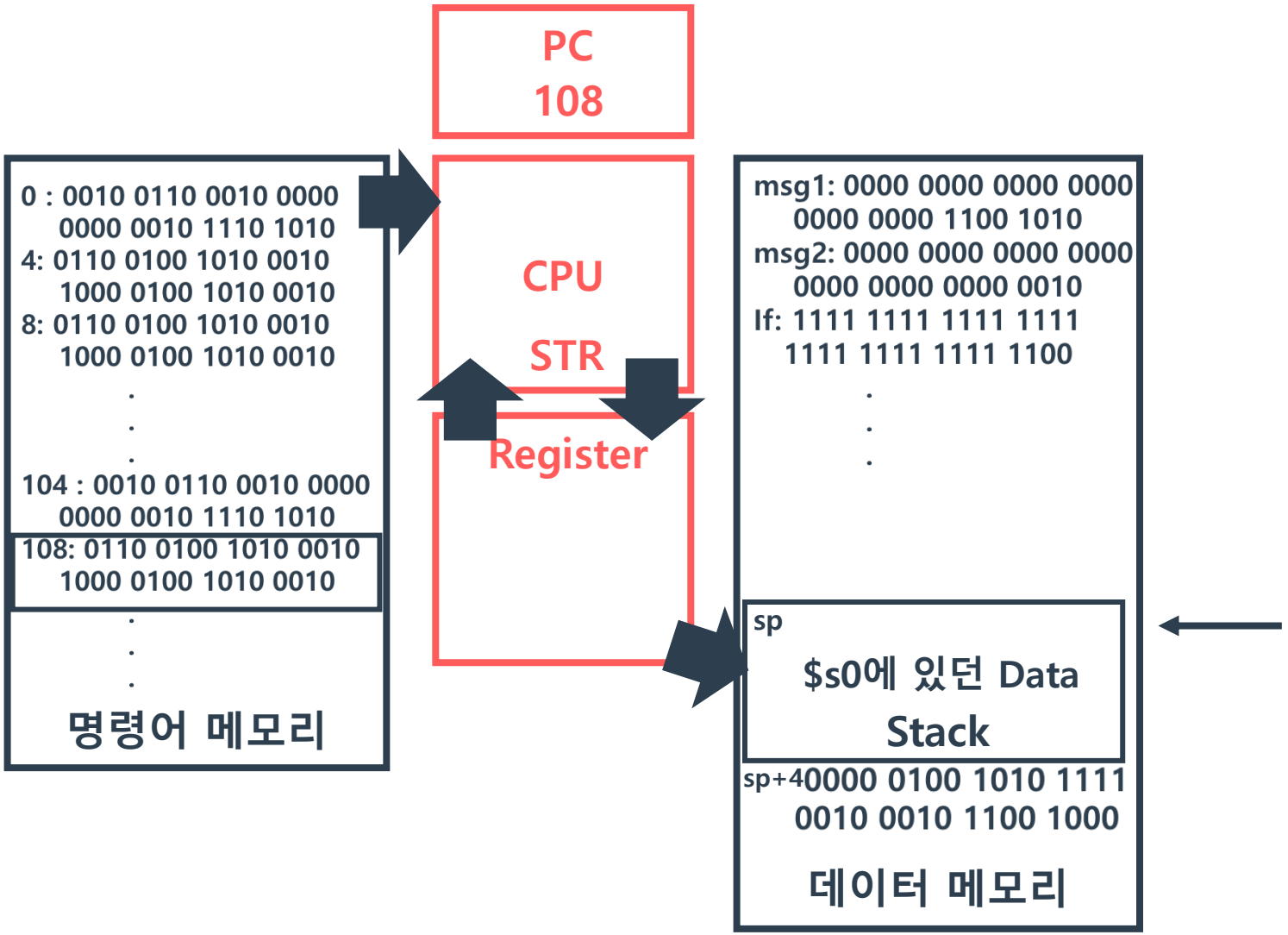
```
# ------------------------------------------------------------------

        # FUNCTION: int add(int num1, int num2)
        # Arguments are stored in $a0 and $a1
        # Return value is stored in $v0
        # Return address is stored in $ra (put there by jal instruction)
        # Typical function operation is:
        #  1.) Store registers on the stack that we will overwrite
        #  2.) Run the function
        #  3.) Save the return value
        #  4.) Restore registers from the stack
        #  5.) Return (jump) to previous location
        # Note: This function is longer than it needs to be,
        # in order to demonstrate the usual 5 step function process...

add2:   # Store registers on the stack that we will overwrite (just $s0)
104     addi $sp,$sp, -4        # Adjust stack pointer
108     sw $s0,0($sp)           # Save $s0 on the stack

        # Run the function
112     add $s0,$a0,$a1         # Sum = sum + i

        # Save the return value in $v0
116     move $v0,$s0

        # Restore overwritten registers from the stack
120     lw $s0,0($sp)
124     addi $sp,$sp,4          # Adjust stack pointer

        # Return from function
128     jr $ra                  # Jump to addr stored in $ra

# ------------------------------------------------------------------
```

```
        # Start .data segment (data!)
        .data
132 msg1:   .asciiz "Number of integers (N)?  "
136 msg2:   .asciiz "Sum = "
140 lf:     .asciiz "\n"
```

General Execution Flow (Cont'd)



General Execution Flow (Cont'd)



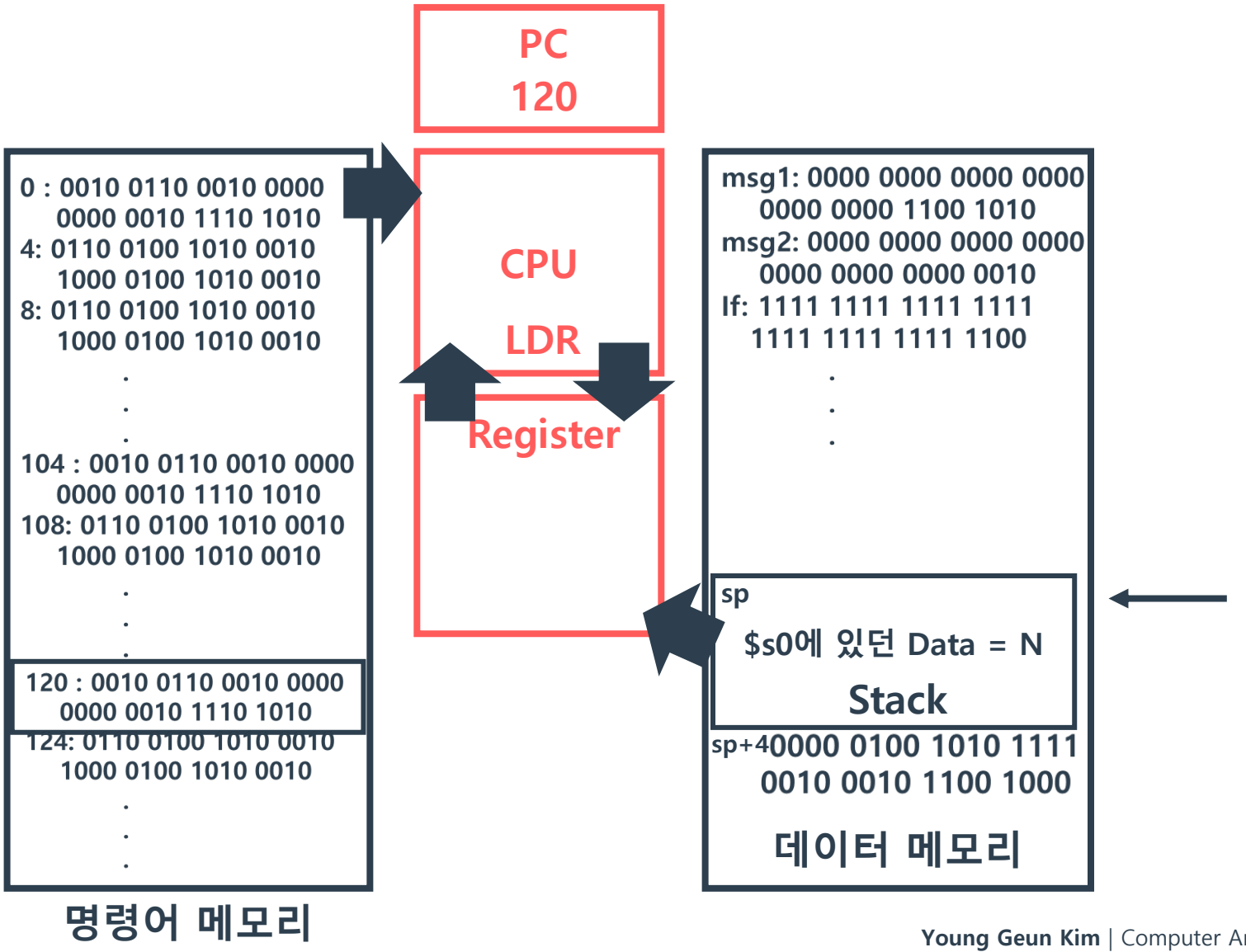
Procedure Call

```
                # ------------------------------------------------------------------

                        .text

                        .globl  main
main:
                # Register assignments
                # $s0 = N
                # $s1 = counter (i)
                # $s2 = sum

                # Print msg1
0               li      $v0,4           # print_string syscall code = 4
4               la      $a0, msg1
8               syscall

                # Get N from user and save
12              li      $v0,5           # read_int syscall code = 5
16              syscall                 = N
20              move    $s0,$v0         # syscall results returned in $v0

                # Initialize registers
24              li      $s1, 0          # Reg $s1 = counter (i)
28              li      $s2, 0          # Reg $s2 = sum

                # Main loop body
32  loop:       addi    $s1, $s1, 1     # i = i + 1

                # Call add function
36              move    $a0, $s2        # Argument 1: sum ($s2)
40              move    $a1, $s1        # Argument 2: i ($s1)
44              jal     add2    = 104   # Save current PC in $ra, and jump to add2
48              move    $s2,$v0         # Return value saved in $v0. This is sum ($s2)
52              beq     $s0, $s1, exit  # if i = N, continue
56              j       loop

                # Exit routine - print msg2
60  exit:       li      $v0, 4          # print_string syscall code = 4
64              la      $a0, msg2
68              syscall

                # Print sum
72              li      $v0,1           # print_string syscall code = 4
76              move    $a0, $s2
80              syscall

                # Print newline
84              li      $v0,4           # print_string syscall code = 4
88              la      $a0, lf
92              syscall
96              li      $v0,10          # exit
100             syscall
```

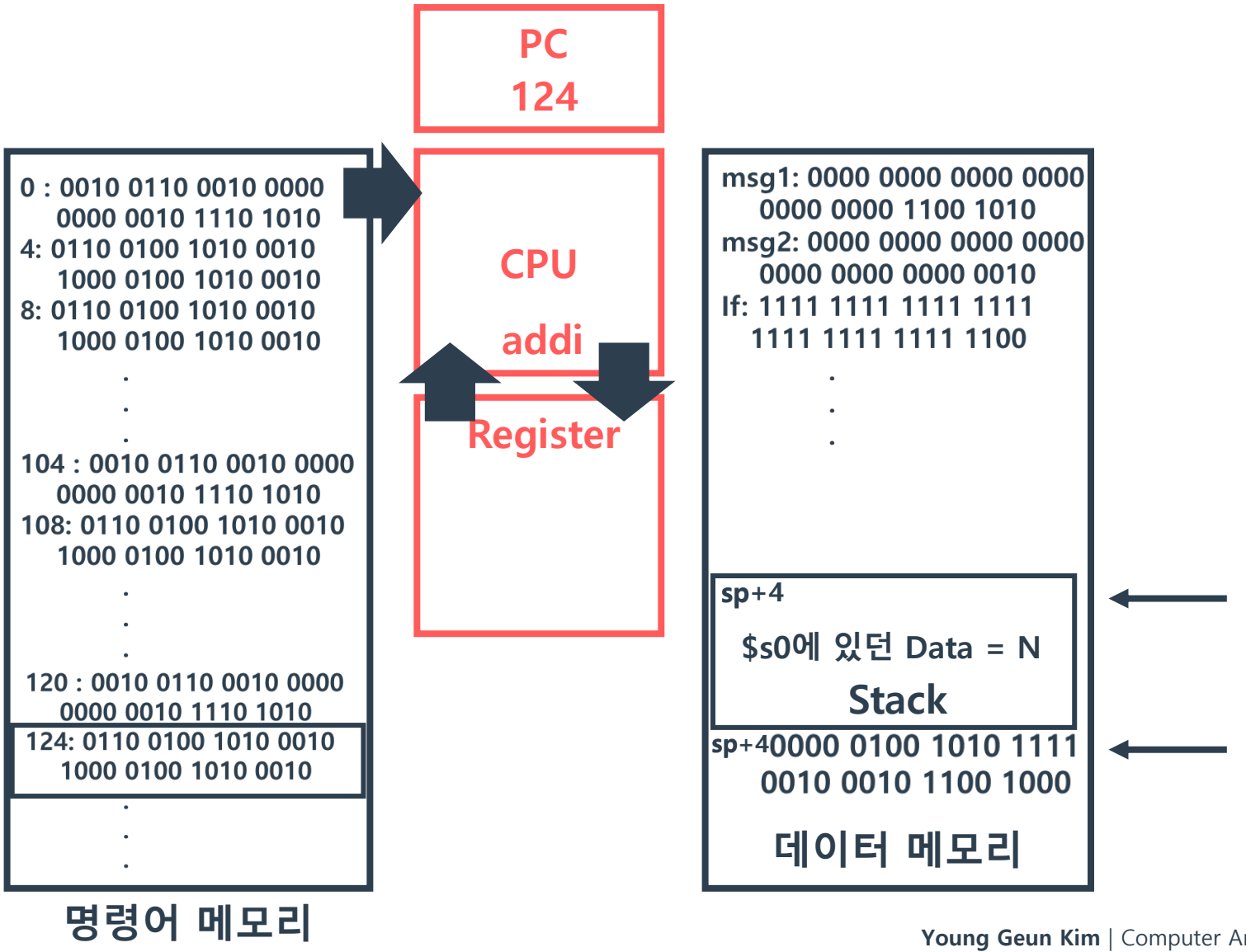
```
                # ------------------------------------------------------------------

                        # FUNCTION: int add(int num1, int num2)
                        # Arguments are stored in $a0 and $a1
                        # Return value is stored in $v0
                        # Return address is stored in $ra (put there by jal instruction)
                        # Typical function operation is:
                        #  1.) Store registers on the stack that we will overwrite
                        #  2.) Run the function
                        #  3.) Save the return value
                        #  4.) Restore registers from the stack
                        #  5.) Return (jump) to previous location
                        # Note: This function is longer than it needs to be,
                        # in order to demonstrate the usual 5 step function process...

                add2:   # Store registers on the stack that we will overwrite (just $s0)
104                     addi $sp,$sp, -4        # Adjust stack pointer
108                     sw $s0,0($sp)           # Save $s0 on the stack

                        # Run the function
112                     add $s0,$a0,$a1         # Sum = sum + i

                        # Save the return value in $v0
116                     move $v0,$s0

                        # Restore overwritten registers from the stack
120                     lw $s0,0($sp)
124                     addi $sp,$sp,4          # Adjust stack pointer

                        # Return from function
128                     jr $ra                  # Jump to addr stored in $ra

                # ------------------------------------------------------------------
```

```
                        # Start .data segment (data!)
                        .data
132             msg1:   .asciiz "Number of integers (N)?  "
136             msg2:   .asciiz "Sum = "
140             lf:     .asciiz "\n"
```

General Execution Flow (Cont'd)



General Execution Flow (Cont'd)



Procedure Call

```
                # ------------------------------------------------------------------

                        .text

                        .globl  main
main:
                        # Register assignments
                        # $s0 = N
                        # $s1 = counter (i)
                        # $s2 = sum

                        # Print msg1
0                       li      $v0,4           # print_string syscall code = 4
4                       la      $a0, msg1
8                       syscall

                        # Get N from user and save
12                      li      $v0,5           # read_int syscall code = 5
16                      syscall                 = N
20                      move    $s0,$v0         # syscall results returned in $v0

                        # Initialize registers
24                      li i =  $s1  0          # Reg $s1 = counter (i)
28                      li      $s2, 0          # Reg $s2 = sum

                        # Main loop body
32      loop:           addi    $s1, $s1, 1     # i = i + 1

                        # Call add function
36                      move    $a0, $s2        # Argument 1: sum ($s2)
40                      move    $a1, $s1        # Argument 2: i ($s1)
44                      jal     add2    = 104   # Save current PC in $ra, and jump to add2
48                      move    $s2,$v0         # Return value saved in $v0. This is sum ($s2)
52                      beq     $s0, $s1, exit  # if i = N, continue
56                      j       loop    = 32

                        # Exit routine - print msg2
60      exit:           li      $v0, 4          # print_string syscall code = 4
64                      la      $a0, msg2
68                      syscall

                        # Print sum
72                      li      $v0,1           # print_string syscall code = 4
76                      move    $a0, $s2
80                      syscall

                        # Print newline
84                      li      $v0,4           # print_string syscall code = 4
88                      la      $a0, lf
92                      syscall
96                      li      $v0,10          # exit
100                     syscall
```

```
                # ------------------------------------------------------------------

                        # FUNCTION: int add(int num1, int num2)
                        # Arguments are stored in $a0 and $a1
                        # Return value is stored in $v0
                        # Return address is stored in $ra (put there by jal instruction)
                        # Typical function operation is:
                        #  1.) Store registers on the stack that we will overwrite
                        #  2.) Run the function
                        #  3.) Save the return value
                        #  4.) Restore registers from the stack
                        #  5.) Return (jump) to previous location
                        # Note: This function is longer than it needs to be,
                        # in order to demonstrate the usual 5 step function process...

104     add2:   # Store registers on the stack that we will overwrite (just $s0)
                addi $sp,$sp, -4        # Adjust stack pointer
108             sw $s0,0($sp)           # Save $s0 on the stack

                # Run the function
112             add $s0,$a0,$a1         # Sum = sum + i

                # Save the return value in $v0
116             move $v0,$s0

                # Restore overwritten registers from the stack
120             lw $s0,0($sp)
124             addi $sp,$sp,4          # Adjust stack pointer

                # Return from function
128             jr $ra          = 48    # Jump to addr stored in $ra

                # ------------------------------------------------------------------
```

```
                # Start .data segment (data!)
                .data
132     msg1:   .asciiz "Number of integers (N)?  "
136     msg2:   .asciiz "Sum = "
140     lf:     .asciiz "\n"
```

Procedure Call

```
0   # ------------------------------------------------------------------
4           .text
8           .globl  main
    main:
                # Register assignments
                # $s0 = N
                # $s1 = counter (i)
                # $s2 = sum

                # Print msg1
0               li      $v0,4           # print_string syscall code = 4
4               la      $a0, msg1
8               syscall

                # Get N from user and save
12              li      $v0,5           # read_int syscall code = 5
16              syscall                 = N
20              move    $s0,$v0         # syscall results returned in $v0

                # Initialize registers
24              li i =  $s1, 0          # Reg $s1 = counter (i)
28              li      $s2, 0          # Reg $s2 = sum

                # Main loop body
32  loop:       addi    $s1, $s1, 1     # i = i + 1

                # Call add function
36              move    $a0, $s2        # Argument 1: sum ($s2)
40              move    $a1, $s1        # Argument 2: i ($s1)
44              jal     add2    = 104   # Save current PC in $ra, and jump to add2
48              move    $s2,$v0         # Return value saved in $v0. This is sum ($s2)
52              beq     $s0, $s1, exit  # if i = N, continue
56              j       loop            = 60

                # Exit routine - print msg2
60  exit:       li      $v0, 4          # print_string syscall code = 4
64              la      $a0, msg2
68              syscall

                # Print sum
72              li      $v0,1           # print_string syscall code = 4
76              move    $a0, $s2
80              syscall

                # Print newline
84              li      $v0,4           # print_string syscall code = 4
88              la      $a0, lf
92              syscall
96              li      $v0,10          # exit
100             syscall
```

```
    # ------------------------------------------------------------------
            # FUNCTION: int add(int num1, int num2)
            # Arguments are stored in $a0 and $a1
            # Return value is stored in $v0
            # Return address is stored in $ra (put there by jal instruction)
            # Typical function operation is:
            #  1.) Store registers on the stack that we will overwrite
            #  2.) Run the function
            #  3.) Save the return value
            #  4.) Restore registers from the stack
            #  5.) Return (jump) to previous location
            # Note: This function is longer than it needs to be,
            # in order to demonstrate the usual 5 step function process...

    add2:   # Store registers on the stack that we will overwrite (just $s0)
104             addi $sp,$sp, -4        # Adjust stack pointer
108             sw $s0,0($sp)           # Save $s0 on the stack

                # Run the function
112             add $s0,$a0,$a1         # Sum = sum + i

                # Save the return value in $v0
116             move $v0,$s0

                # Restore overwritten registers from the stack
120             lw $s0,0($sp)
124             addi $sp,$sp,4          # Adjust stack pointer

                # Return from function
128             jr $ra          = 48    # Jump to addr stored in $ra
    # ------------------------------------------------------------------
```

```
            # Start .data segment (data!)
            .data
132 msg1:   .asciiz "Number of integers (N)?  "
136 msg2:   .asciiz "Sum = "
140 lf:     .asciiz "\n"
```

Procedure Call

```
# ---------------------------------------------------------------

        .text

        .globl  main
main:
        # Register assignments
        # $s0 = N
        # $s1 = counter (i)
        # $s2 = sum

        # Print msg1
        li      $v0,4           # print_string syscall code = 4
        la      $a0, msg1
        syscall

        # Get N from user and save
        li      $v0,5           # read_int syscall code = 5
        syscall
        move    $s0,$v0         # syscall results returned in $v0

        # Initialize registers
        li      $s1, 0          # Reg $s1 = counter (i)
        li      $s2, 0          # Reg $s2 = sum

        # Main loop body
loop:   addi    $s1, $s1, 1     # i = i + 1

        # Call add function
        move    $a0, $s2        # Argument 1: sum ($s2)
        move    $a1, $s1        # Argument 2: i ($s1)
        jal     add2            # Save current PC in $ra, and jump to add2
        move    $s2,$v0         # Return value saved in $v0. This is sum ($s2)
        beq     $s0, $s1, exit  # if i = N, continue
        j       loop

        # Exit routine - print msg2
exit:   li      $v0, 4          # print_string syscall code = 4
        la      $a0, msg2
        syscall

        # Print sum
        li      $v0,1           # print_string syscall code = 4
        move    $a0, $s2
        syscall

        # Print newline
        li      $v0,4           # print_string syscall code = 4
        la      $a0, lf
        syscall
        li      $v0,10          # exit
        syscall
```

```
# ---------------------------------------------------------------

        # FUNCTION: int add(int num1, int num2)
        # Arguments are stored in $a0 and $a1
        # Return value is stored in $v0
        # Return address is stored in $ra (put there by jal instruction)
        # Typical function operation is:
        #  1.) Store registers on the stack that we will overwrite
        #  2.) Run the function
        #  3.) Save the return value
        #  4.) Restore registers from the stack
        #  5.) Return (jump) to previous location
        # Note: This function is longer than it needs to be,
        # in order to demonstrate the usual 5 step function process...

add2:   # Store registers on the stack that we will overwrite (just $s0)
        addi $sp,$sp, -4        # Adjust stack pointer
        sw $s0,0($sp)           # Save $s0 on the stack

        # Run the function
        add $s0,$a0,$a1         # Sum = sum + i

        # Save the return value in $v0
        move $v0,$s0

        # Restore overwritten registers from the stack
        lw $s0,0($sp)
        addi $sp,$sp,4          # Adjust stack pointer

        # Return from function
        jr $ra                  # Jump to addr stored in $ra

# ---------------------------------------------------------------
```

```
        # Start .data segment (data!)
        .data
msg1:   .asciiz "Number of integers (N)?  "
msg2:   .asciiz "Sum = "
lf:     .asciiz "\n"
```

Procedure Call

또 다른 참고 블로그: https://gusdnd852.tistory.com/245