p과제 4

마이크로프로세서응용

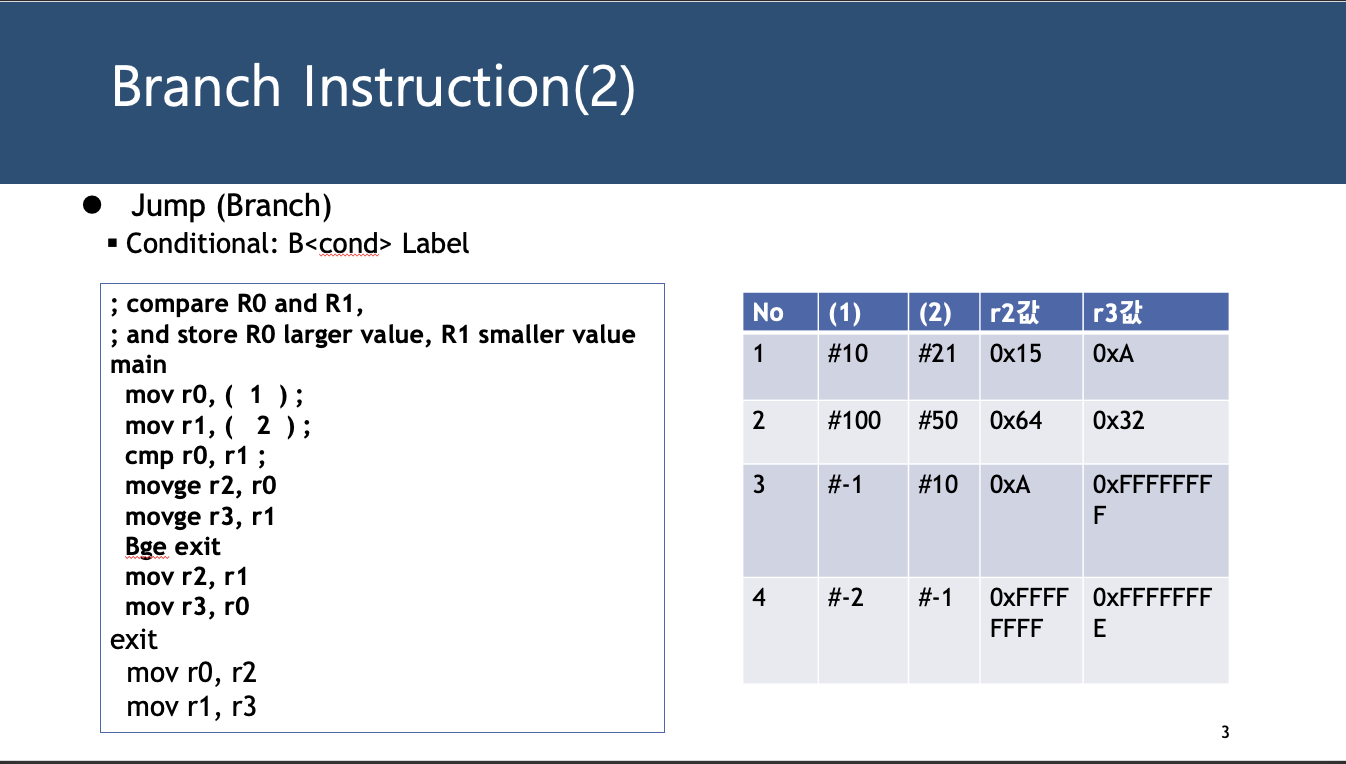
22100113 김성빈

Lab workbook Assembly 3

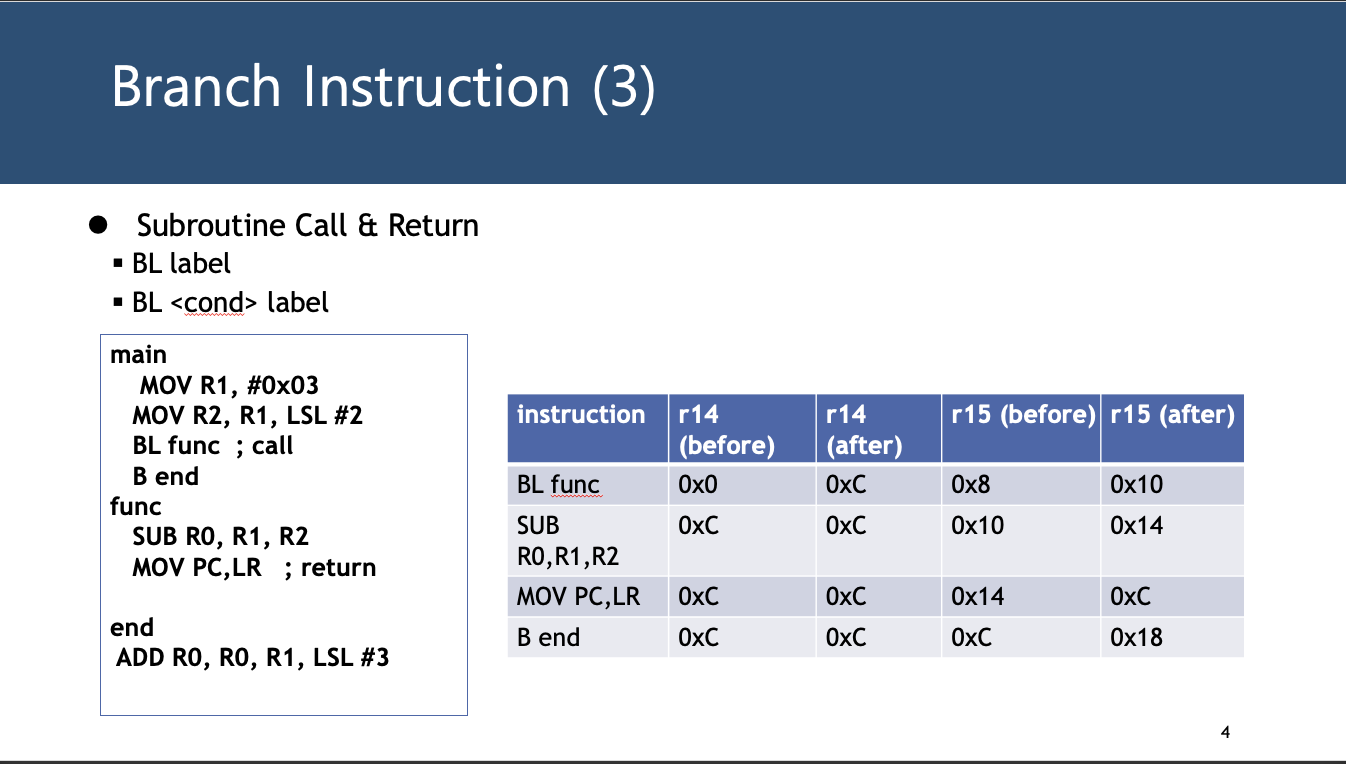
A blue and white instruction page

Description automatically generated with medium confidence

[Branch Instruction (1)]



[Branch Instruction (2)]



[Branch Instruction (3)]

A diagram of a function

Description automatically generated

[Branch Instruction (4-1)]

A blue and white instruction table

Description automatically generated

[Branch Instruction (4-2)]

A screenshot of a computer

Description automatically generated

[Exercise-1]

COUNT\_EVEN

BIC R4, R4, #0xFFFFFFFF

AND R4, R0, #0x1

CMP R4, #0x0

ADDEQ R3, R3, #1

BIC R4, R4, #0xFFFFFFFF

AND R4, R1, #0x1

CMP R4, #0x0

ADDEQ R3, R3, #1

BIC R4, R4, #0xFFFFFFFF

AND R4, R2, #0x1

CMP R4, #0x0

ADDEQ R3, R3, #1

MOV PC, LR

[Exercise-1-code]

A screenshot of a computer

Description automatically generated

[Exercise-2]

A screenshot of a computer program

Description automatically generated

[Exercise-3]

factorial

STMFD SP!, {R0, LR}

CMP R0, #1

BLE less\_than\_two

SUB R0, R0, #1

BL factorial

B done

less\_than\_two

MOV R1, #1  
 LDMFD SP!, {R0, LR}

MOV PC, LR

done

LDMFD SP!, {R0, LR}

MULS R1, R1, R0

MOV PC, LR

[Exercise-3-code]

.section .data

fibo\_cache:

.word -1, -1, -1, -1, -1, -1, -1, -1

.section .text

.global \_start

\_start:

MOV R0, #6

MOV R6, #0x4

LDR R1, =fibo\_cache

BL fibo

B end

fibo:

STMFD SP!, {R0, LR}

MUL R7, R0, R6 // R7 = R0 \* 0x4

LDR R2, [R1, R7] // R2 = R1[R0]

CMP R2, #-1 // check if uninitialized

BNE done // if initialized, return R1[R0]

CMP R0, #1 // base case, n <= 1

MUL R7, R0, R6 // R7 = R0 \* 0x4

STRLE R0, [R1, R7] // R1[R0] = R0

BLE done

MOV R8, R0 // store n for mem[n] = fibo(n-1) + fibo(n-2)

STMFD SP!, {R8}

SUB R0, R0, #1 // fibo(n-1)

BL fibo

MOV R4, R3

SUB R0, R0, #1 // fibo(n-2)

BL fibo

ADD R5, R4, R3 // fibo(n-1) + fibo(n-2)

LDMFD SP!, {R8}

MUL R7, R8, R6 // R7 = R8 \* 0x4

STR R5, [R1, R7] // mem[n] = fibo(n-1) + fibo(n-2)

B done

done: // return mem[n], R3 = mem[n]

LDMFD SP!, {R0, LR}

MUL R7, R0, R6 // R7 = R0 \* 0x4

LDR R3, [R1, R7] // return (R3 = R1[R0]), result stored in R3

MOV PC, LR

end:

[Exercise-4-BONUS-code]