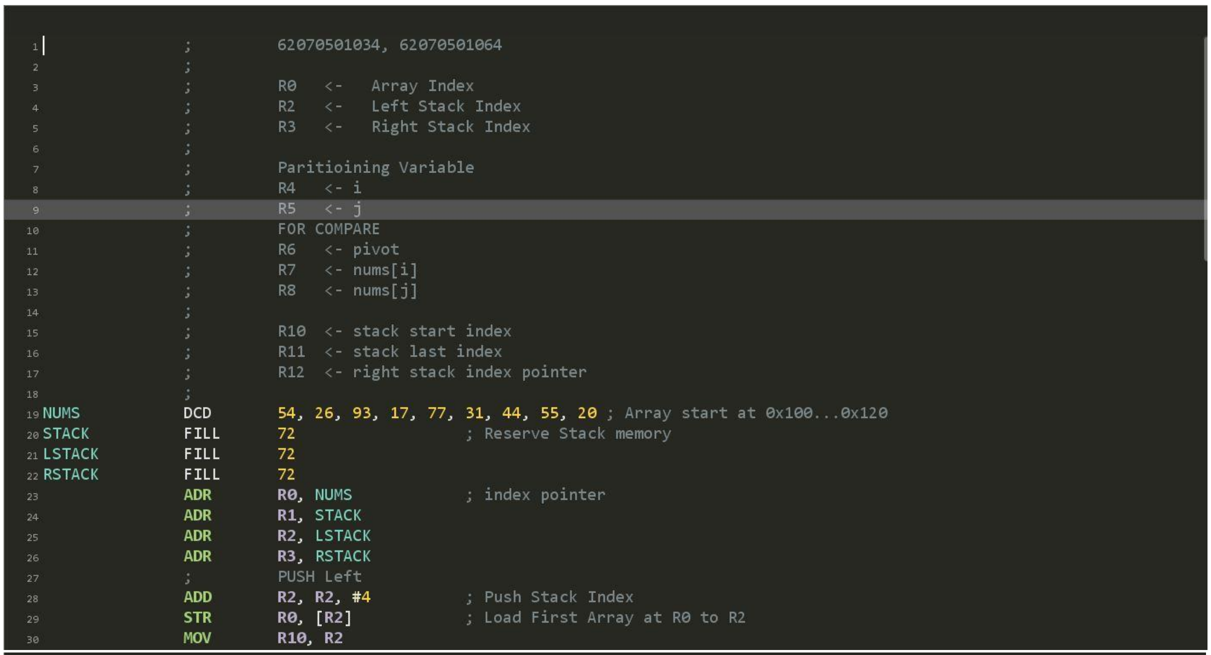
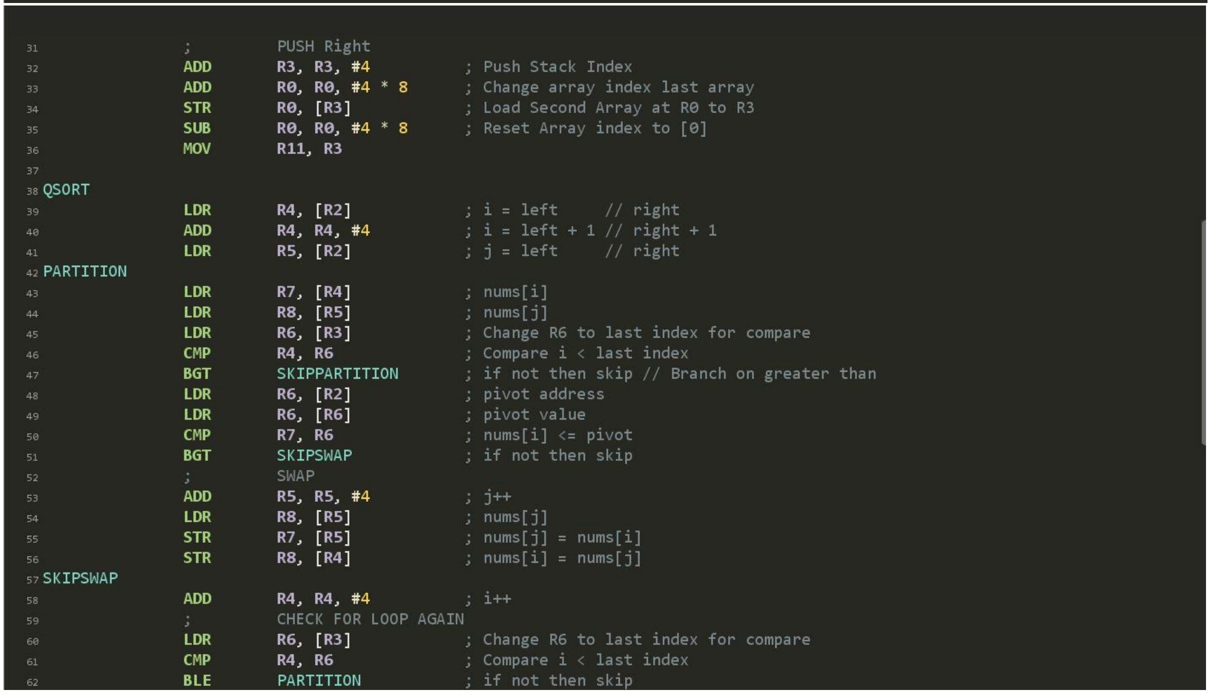
LAB 4 CPE224 Computer Architecture

Member: 62070501034 Nanthakan Rujilakhanon, 62070501064 Onwipa Kujaroenpaisan

Sorting: Quick Sort

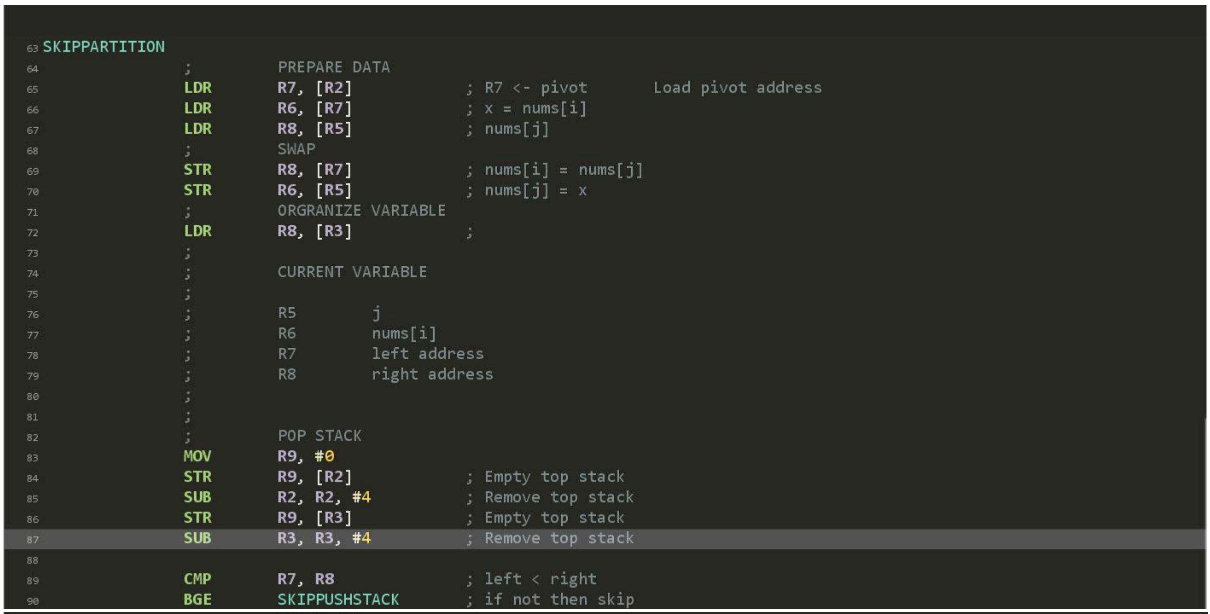


```
               ;         62070501034, 62070501064
               ;
               ;         R0   <-   Array Index
               ;         R2   <-   Left Stack Index
               ;         R3   <-   Right Stack Index
               ;
               ;         Paritioining Variable
               ;         R4   <- i
               ;         R5   <- j
               ;         FOR COMPARE
               ;         R6   <- pivot
               ;         R7   <- nums[i]
               ;         R8   <- nums[j]
               ;
               ;         R10  <- stack start index
               ;         R11  <- stack last index
               ;         R12  <- right stack index pointer
               ;
NUMS           DCD       54, 26, 93, 17, 77, 31, 44, 55, 20 ; Array start at 0x100...0x120
STACK          FILL      72                   ; Reserve Stack memory
LSTACK         FILL      72
RSTACK         FILL      72
               ADR       R0, NUMS             ; index pointer
               ADR       R1, STACK
               ADR       R2, LSTACK
               ADR       R3, RSTACK
               ;         PUSH Left
               ADD       R2, R2, #4           ; Push Stack Index
               STR       R0, [R2]             ; Load First Array at R0 to R2
               MOV       R10, R2
```



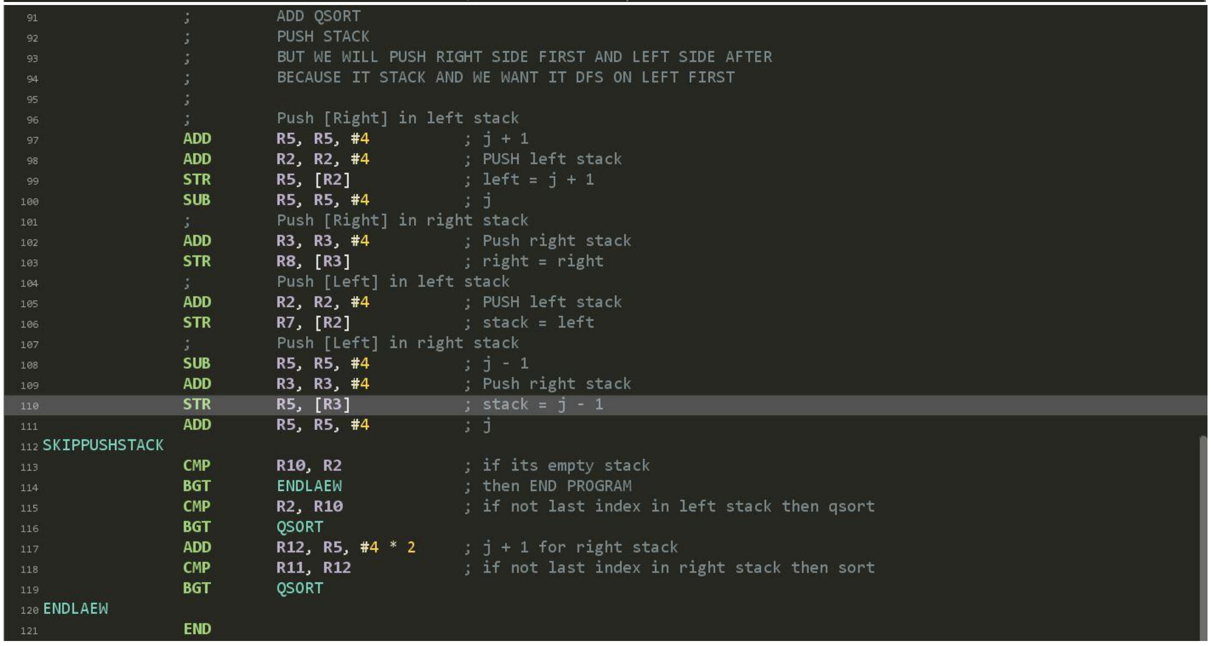
```
               ;         PUSH Right
               ADD       R3, R3, #4           ; Push Stack Index
               ADD       R0, R0, #4 * 8       ; Change array index last array
               STR       R0, [R3]             ; Load Second Array at R0 to R3
               SUB       R0, R0, #4 * 8       ; Reset Array index to [0]
               MOV       R11, R3

QSORT
               LDR       R4, [R2]             ; i = left     // right
               ADD       R4, R4, #4           ; i = left + 1 // right + 1
               LDR       R5, [R2]             ; j = left     // right
PARTITION
               LDR       R7, [R4]             ; nums[i]
               LDR       R8, [R5]             ; nums[j]
               LDR       R6, [R3]             ; Change R6 to last index for compare
               CMP       R4, R6               ; Compare i < last index
               BGT       SKIPPARTITION        ; if not then skip // Branch on greater than
               LDR       R6, [R2]             ; pivot address
               LDR       R6, [R6]             ; pivot value
               CMP       R7, R6               ; nums[i] <= pivot
               BGT       SKIPSWAP             ; if not then skip
               ;         SWAP
               ADD       R5, R5, #4           ; j++
               LDR       R8, [R5]             ; nums[j]
               STR       R7, [R5]             ; nums[j] = nums[i]
               STR       R8, [R4]             ; nums[i] = nums[j]
SKIPSWAP
               ADD       R4, R4, #4           ; i++
               ;         CHECK FOR LOOP AGAIN
               LDR       R6, [R3]             ; Change R6 to last index for compare
               CMP       R4, R6               ; Compare i < last index
               BLE       PARTITION            ; if not then skip
```

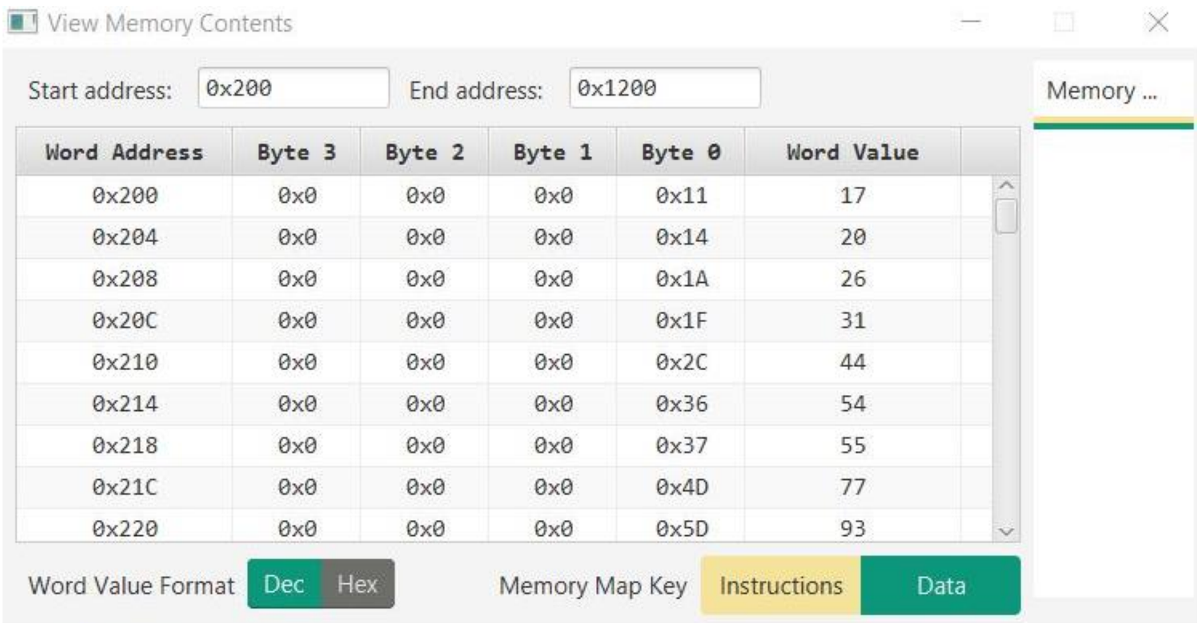


```
SKIPPARTITION
               ;         PREPARE DATA
               LDR       R7, [R2]            ; R7 <- pivot       Load pivot address
               LDR       R6, [R7]            ; x = nums[i]
               LDR       R8, [R5]            ; nums[j]
               ;         SWAP
               STR       R8, [R7]            ; nums[i] = nums[j]
               STR       R6, [R5]            ; nums[j] = x
               ;         ORGRANIZE VARIABLE
               LDR       R8, [R3]            ;
               ;
               ;         CURRENT VARIABLE
               ;
               ;         R5        j
               ;         R6        nums[i]
               ;         R7        left address
               ;         R8        right address
               ;
               ;
               ;         POP STACK
               MOV       R9, #0
               STR       R9, [R2]            ; Empty top stack
               SUB       R2, R2, #4          ; Remove top stack
               STR       R9, [R3]            ; Empty top stack
               SUB       R3, R3, #4          ; Remove top stack

               CMP       R7, R8              ; left < right
               BGE       SKIPPUSHSTACK       ; if not then skip
```



```
               ;         ADD QSORT
               ;         PUSH STACK
               ;         BUT WE WILL PUSH RIGHT SIDE FIRST AND LEFT SIDE AFTER
               ;         BECAUSE IT STACK AND WE WANT IT DFS ON LEFT FIRST
               ;
               ;         Push [Right] in left stack
               ADD       R5, R5, #4          ; j + 1
               ADD       R2, R2, #4          ; PUSH left stack
               STR       R5, [R2]            ; left = j + 1
               SUB       R5, R5, #4          ; j
               ;         Push [Right] in right stack
               ADD       R3, R3, #4          ; Push right stack
               STR       R8, [R3]            ; right = right
               ;         Push [Left] in left stack
               ADD       R2, R2, #4          ; PUSH left stack
               STR       R7, [R2]            ; stack = left
               ;         Push [Left] in right stack
               SUB       R5, R5, #4          ; j - 1
               ADD       R3, R3, #4          ; Push right stack
               STR       R5, [R3]            ; stack = j - 1
               ADD       R5, R5, #4          ; j
SKIPPUSHSTACK
               CMP       R10, R2             ; if its empty stack
               BGT       ENDLAEW             ; then END PROGRAM
               CMP       R2, R10             ; if not last index in left stack then qsort
               BGT       QSORT
               ADD       R12, R5, #4 * 2     ; j + 1 for right stack
               CMP       R11, R12            ; if not last index in right stack then sort
               BGT       QSORT
ENDLAEW
               END
```



View Memory Contents

Start address: 0x200 End address: 0x1200

Word Address	Byte 3	Byte 2	Byte 1	Byte 0	Word Value
0x200	0x0	0x0	0x0	0x11	17
0x204	0x0	0x0	0x0	0x14	20
0x208	0x0	0x0	0x0	0x1A	26
0x20C	0x0	0x0	0x0	0x1F	31
0x210	0x0	0x0	0x0	0x2C	44
0x214	0x0	0x0	0x0	0x36	54
0x218	0x0	0x0	0x0	0x37	55
0x21C	0x0	0x0	0x0	0x4D	77
0x220	0x0	0x0	0x0	0x5D	93

Word Value Format: [✓ Dec] Hex

Memory Map Key: Instructions, Data