Name: Nicolas Thouchkaieff
ID: 50171348

The code is explained with comments.

```
    .data
int_format_string: .asciz " %d"
select_number: .asciz "Select a number (1. Push 2. Pop 3. Stack print 4. Exit): "
type_number_push: .asciz "Type a value to push: "
print_popped_value: .asciz "The popped value is: %d\n"
print_stack1: .asciz "The stack content is:  %d, %d, %d, "
print_stack2: .asciz "%d, %d\n"
print_stack_overflow: .asciz "Stack Overflow\n"
print_stack_empty: .asciz "Stack empty\n"

    .align
menu:   .word 0
top:  .word 0
stack:  .word 0, 0, 0, 0, 0

    .text
    .global main
    .extern printf



main:
    stmfd r13!, {r14}
    ldr r7, =stack
    mov r10, #0 @ Stack counter

loop:
    @ Printf Select number
    ldr r0, =select_number
    bl printf

    @ Scanf input number
    ldr r0, =int_format_string
    ldr r1, =menu
    bl scanf

    ldr r1, =menu
    ldr r0, [r1]

    @ Switch case
    cmp r0, #1
    bleq push

    cmp r0, #2
    bleq pop

    cmp r0, #3
    bleq print

    cmp r0, #4
    bleq exit

    b loop
```

Push subroutine :

```
55  push: @ Push subroutine
56      stmfd r13!, {r0-r9, r14}
57
58      @ Printf type value to push
59      ldr r0, =type_number_push
60      bl printf
61
62      @ Scanf input number
63      ldr r0, =int_format_string
64      ldr r1, =top
65      bl scanf
66
67      @ Check if size stack eq 5
68      cmp r10, #5
69      bleq overflow
70
71      @ Save in top
72      ldr r1, =top
73      ldr r0, [r1]
74
75      @ Move elems in stack
76      ldr r1, [r7, #12]
77      str r1, [r7, #16]
78
79      ldr r1, [r7, #8]
80      str r1, [r7, #12]
81
82      ldr r1, [r7, #4]
83      str r1, [r7, #8]
84
85      ldr r1, [r7]
86      str r1, [r7, #4]
87
88      @ Add to the top of stack
89      str r0, [r7]
90
91      @ Increment stack counter
92      add r10, r10, #1
93
94      ldmfd r13!, {r0-r9, pc}
```

Pop subroutine:

```
96   pop: @ Pop subroutine
97       stmfd r13!, {r0-r9, r14}
98
99       @ Check if size stack eq 0
100      cmp r10, #0
101      bleq empty
102
103      @ Printf popped value
104      ldr r0, =print_popped_value
105      ldr r1, [r7]
106      bl printf
107
108      @ Move elems in stack
109      ldr r1, [r7, #4]
110      str r1, [r7]
111
112      ldr r1, [r7, #8]
113      str r1, [r7, #4]
114
115      ldr r1, [r7, #12]
116      str r1, [r7, #8]
117
118      ldr r1, [r7, #16]
119      str r1, [r7, #12]
120
121      mov r1, #0
122      str r1, [r7, #16]
123
124      @ Decrement stack counter
125      sub r10, r10, #1
126
127      ldmfd r13!, {r0-r9, pc}
```

```
129  print: @ Print stack subroutine
130      stmfd r13!, {r0-r12, r14}
131
132      ldr r0, =print_stack1
133      ldr r1, [r7]
134      ldr r2, [r7, #4]
135      ldr r3, [r7, #8]
136      bl printf
137
138      ldr r0, =print_stack2
139      ldr r1, [r7, #12]
140      ldr r2, [r7, #16]
141      bl printf
142      ldmfd r13!, {r0-r12, pc}
143
144  overflow: @ Print stack overflow
145      stmfd r13!, {r0-r12, r14}
146
147      ldr r0, =print_stack_overflow
148      bl printf
149      bl loop
150
151      ldmfd r13!, {r0-r12, pc}
152
153  empty: @ Print stack empty
154      stmfd r13!, {r0-r12, r14}
155
156      ldr r0, =print_stack_empty
157      bl printf
158      bl loop
159
160      ldmfd r13!, {r0-r12, pc}
161
162  exit:
163      ldmfd r13!, {pc}
```