Name: Nicolas Thouchkaieff
ID: 50171348

The code is explained with comments

```
1       .data
2    int_format_string: .asciz " %d"
3    select_number: .asciz "Select a number (1. Push 2. Pop 3. Stack print 4. Exit): "
4    type_number_push: .asciz "Type a value to push: "
5    print_popped_value: .asciz "The popped value is: %d\n"
6    print_stack1: .asciz "The stack content is:  %d, %d, %d, "
7    print_stack2: .asciz "%d, %d\n"
8    print_stack_overflow: .asciz "Stack Overflow\n"
9    print_stack_empty: .asciz "Stack empty\n"
10
11       .align
12   menu:   .word 0
13   top:  .word 0
14   stack:  .word 0, 0, 0, 0, 0
15
16       .text
17       .global main
18       .extern printf
19
20
21
22   main:
23       stmfd r13!, {r14}
24       ldr r7, =stack
25       mov r10, #0 @ Stack counter
26
27   loop:
28       @ Printf Select number
29       ldr r0, =select_number
30       bl printf
31
32       @ Scanf input number
33       ldr r0, =int_format_string
34       ldr r1, =menu
35       bl scanf
36
37       ldr r1, =menu
38       ldr r0, [r1]
39
40       @ Switch case
41       cmp r0, #1
42       bleq push
43
44       cmp r0, #2
45       bleq pop
46
47       cmp r0, #3
48       bleq print
49
50       cmp r0, #4
51       bleq exit
52
53       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}
```

```
print: @ Print stack subroutine
    stmfd r13!, {r0-r12, r14}

    ldr r0, =print_stack1
    ldr r1, [r7]
    ldr r2, [r7, #4]
    ldr r3, [r7, #8]
    bl printf

    ldr r0, =print_stack2
    ldr r1, [r7, #12]
    ldr r2, [r7, #16]
    bl printf
    ldmfd r13!, {r0-r12, pc}

overflow: @ Print stack overflow
    stmfd r13!, {r0-r12, r14}

    ldr r0, =print_stack_overflow
    bl printf
    bl loop

    ldmfd r13!, {r0-r12, pc}

empty: @ Print stack empty
    stmfd r13!, {r0-r12, r14}

    ldr r0, =print_stack_empty
    bl printf
    bl loop

    ldmfd r13!, {r0-r12, pc}

exit:
    ldmfd r13!, {pc}
```