SP HW3 Report

a.

Stack:

```
(gdb) i stack
#0  funct_4 (name=4) at hw3_2.c:406
#1  0x000055555555684f in funct_5 (name=5) at hw3_2.c:474
#2  0x00005555555561ee in funct_3 (name=3) at hw3_2.c:350
#3  0x000055555555683a in funct_5 (name=5) at hw3_2.c:472
#4  0x0000555555555ebd in funct_2 (name=2) at hw3_2.c:290
#5  0x0000555555556825 in funct_5 (name=5) at hw3_2.c:470
#6  0x0000555555555b8c in funct_1 (name=1) at hw3_2.c:230
#7  0x0000555555556810 in funct_5 (name=5) at hw3_2.c:468
#8  0x0000555555555233 in main (argc=5, argv=0x7fffffffdf98) at hw3_2.c:114
```

funct1~4的stack pointer(rsp) and base pointer(rbp)

```
Breakpoint 5, funct_1 (name=1) at hw3_2.c:226
226     iii+1];
(gdb) i reg
rax            0x0      0
rbx            0x0      0
rcx            0x555555764710   93824994395920
rdx            0x690ddbd9ad3d5511       7569948276640535825
rsi            0x0      0
rdi            0x1      1
rbp            0x7fffffff3fb0   0x7fffffff3fb0
rsp            0x7fffffff3e80   0x7fffffff3e80
Breakpoint 6, funct_2 (name=2) at hw3_2.c:286
286     iii+1];
(gdb) i reg
rax            0x1      1
rbx            0x0      0
rcx            0x555555764710   93824994395920
rdx            0x690ddbd8685d5511       7569948271190037777
rsi            0x0      0
rdi            0x2      2
rbp            0x7fffffffea200  0x7fffffffea200
rsp            0x7fffffffea0d0  0x7fffffffea0d0
```

```
Breakpoint 7, funct_3 (name=3) at hw3_2.c:346
346     [iii+1];
(gdb) i reg
rax            0x2      2
rbx            0x0      0
rcx            0x555555764710   93824994395920
rdx            0x690ddbdb54fd5511       7569948283749881105
rsi            0x0      0
rdi            0x3      3
rbp            0x7ffffffe0450   0x7ffffffe0450
rsp            0x7ffffffe0320   0x7ffffffe0320
Breakpoint 4, funct_4 (name=4) at hw3_2.c:406
406     ;
(gdb) i reg
rax            0x3      3
rbx            0x0      0
rcx            0x555555764710   93824994395920
rdx            0x690ddbda131d5511       7569948278349714705
rsi            0x0      0
rdi            0x4      4
rbp            0x7ffffffd66a0   0x7ffffffd66a0
rsp            0x7ffffffd6570   0x7ffffffd6570
```

b.

Yes,因為區域變數會被存在funct1~4的Environment裡，每次回來都會回復。

c.

用來避免funct1~3被壓到，這樣就只會壓掉dummy。

d.

Can't,會在funct_3->dummy時crash，因為funct3上面的dummy的一部分被壓掉了。

e.

由於只有task2需要支援第五個參數帶來的操作，而funct1~4是三個task共用的，於是將task1和task3時的第五個參數設為和P相同的值，讓funct1~4能相容task1~3。
在funct1~4使用setjmp和longjmp，讓scheduler可以跳到他們。
使用全域變數cnt1~4來紀錄每個funct分別跑過幾個small loop，因為每次跳回funct時，j都會被設為0，所以需要額外的變數來紀錄之前做了幾次。