Q1)

```
ctxsw:
        push    {r0-r11, lr}            /* Push regs 0 - 11 and lr       */
        push    {lr}                    /* Push return address           */
        mrs     r2, cpsr                /* Obtain status from coprocess.*/
        push    {r2}                    /*   and push onto stack         */
        str     sp, [r0]        /* Save old process's SP        */
        ldr     sp, [r1]        /* Pick up new process's SP   */
        pop     {r0}                    /* Use status as argument and            */
        bl      restore                 /*   call restore to restore it    */
        pop     {lr}                    /* Pick up the return address  */
        pop     {r0-r12}                /* Restore other registere       */
        mov     pc, r12                 /* Return to the new process  */
```

Array implementation:

Pseudocode:

A1,A2: Array of process table to hold contents of registers

```
str r0,[A1] //store register r0 in array
add A1,A1,#4
str r1,[A1]
add A1,A1,#4
str r2,[A1]
add A1,A1,#4
str r3,[A1]
add A1,A1,#4
str r4,[A1]
add A1,A1,#4
str r5,[A1]
add A1,A1,#4
str r6,[A1]
add A1,A1,#4
str r7,[A1]
add A1,A1,#4
str r8,[A1]
add A1,A1,#4
str r9,[A1]
add A1,A1,#4
str r10,[A1]
add A1,A1,#4
str r11,[A1]
add A1,A1,#4
str lr,[A1]
add A1,A1,#4
```

```
str lr,[A1]
add A1,A1,#4
str cpsr,[A1]
add A1,A1,#4
str sp, [r0]
ldr sp, [r1]
sub A1,A1,#64
ldr r0,[A1]
bl restore  //intR.s (system/intR.s)

ldr r0,[A2] //load value into r0 from address pointed by array A2
add A2,A2,#4
ldr r1,[A2]
add A2,A2,#4
ldr r2,[A2]
add A2,A2,#4
ldr r3,[A2]
add A2,A2,#4
ldr r4,[A2]
add A2,A2,#4
ldr r5,[A2]
add A2,A2,#4
ldr r6,[A2]
add A2,A2,#4
ldr r7,[A2]
add A2,A2,#4
ldr r8,[A2]
add A2,A2,#4
ldr r9,[A2]
add A2,A2,#4
ldr r10,[A2]
add A2,A2,#4
ldr r11,[A2]
add A2,A2,#4
ldr lr,[A2]
add A2,A2,#4
ldr pc,[A2]
add A2,A2,#4
```

```
For all registers r0-r11, store its values in array A
Append return address value in array A
Append status register value in array A
Store stack pointer in register r0
Load stack pointer from  register r1
pop stack pointer stored in register r0
bl restore
#context switch
Store all values of array A in r0-r11,program counter and load register
```

Advantages/disadvantages of each approach explained:
Speed:Implementation via arrays will be slower than using stack.
Size of code :Code will be bigger in array implementation.
Memory:Memory required for both the implementation is same.
Stack grows dynamically so there is problem of stack crashing while we use stack.

Q2)
New state PR_DYING introduced in process.h
The call to freeing stack memory in kill.c is removed.Now it will only change the state from PR_FREE to PR_DYING when kill is called.
In create.c,while searching for new process id, it searches for process having PR_DYING state, then deallocates the stack and changes it state to PR_FREE.

Q3)
Resume returns the priority of the process that has been resumed.In case of resume care must be taken to record the priority before calling ready because the resumed process may have higher priority than the currently executing process.If an arbitrary delay can occur between the time resume calls ready and execution continues after the call.During the delay, an arbitrary number of other processes can execute and processes may terminate.Thus to ensure that the returned priority reflects the resumed process's priority at the time of resumption,resume make a copy in local variable prio before calling ready.