Chapter 3
Kernel Structure

System & Network Lab

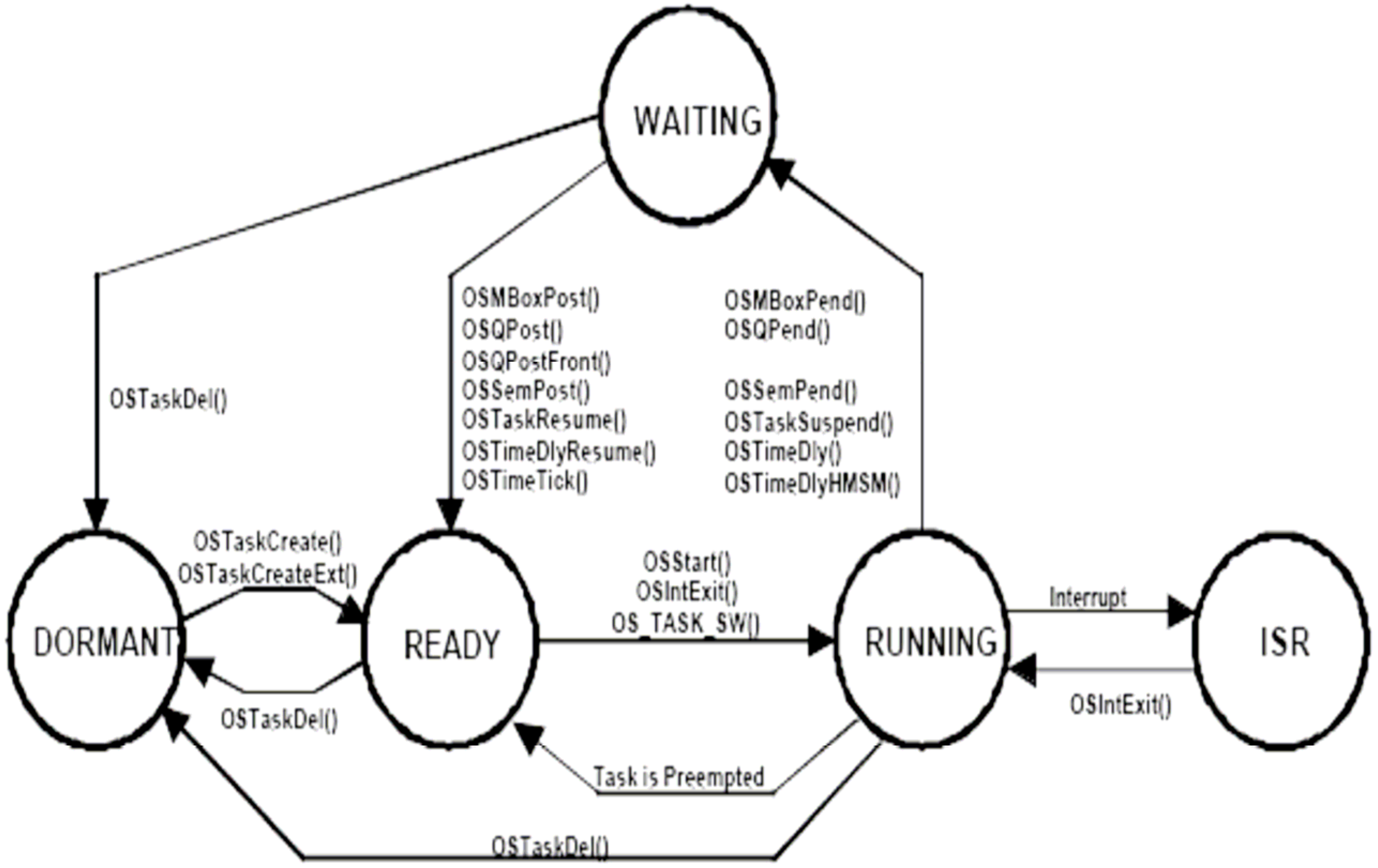
Tsung-Yu Ye

MicroC/OS File Structure

- Processor Specific code
- Processor Independent code – OS system service
- Application Specific (OS_CFG.H , INCLUDES.H)

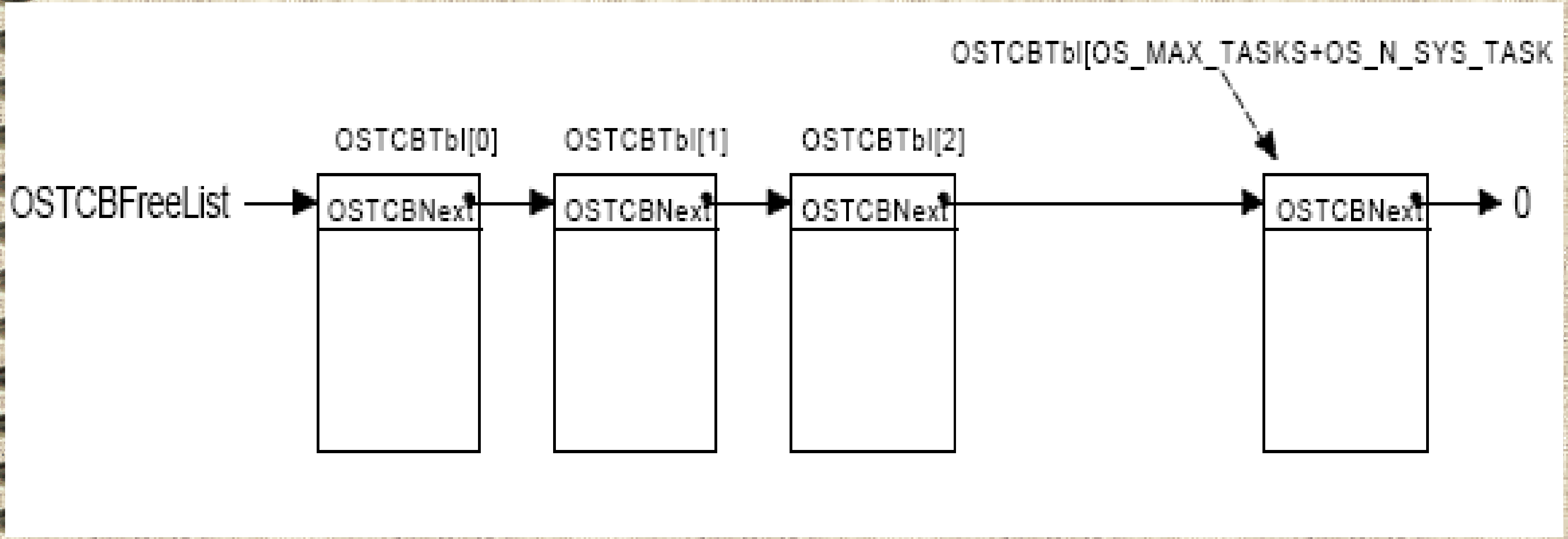
```
void main (void)
{
    OSInit();           /* Initialize uC/OS-II                              */

   /* Create at least 1 task using either OSTaskCreate() or
OSTaskCreateExt(); */

    OSStart();          /* Start multitasking!  OSStart() will not
return */
}
```

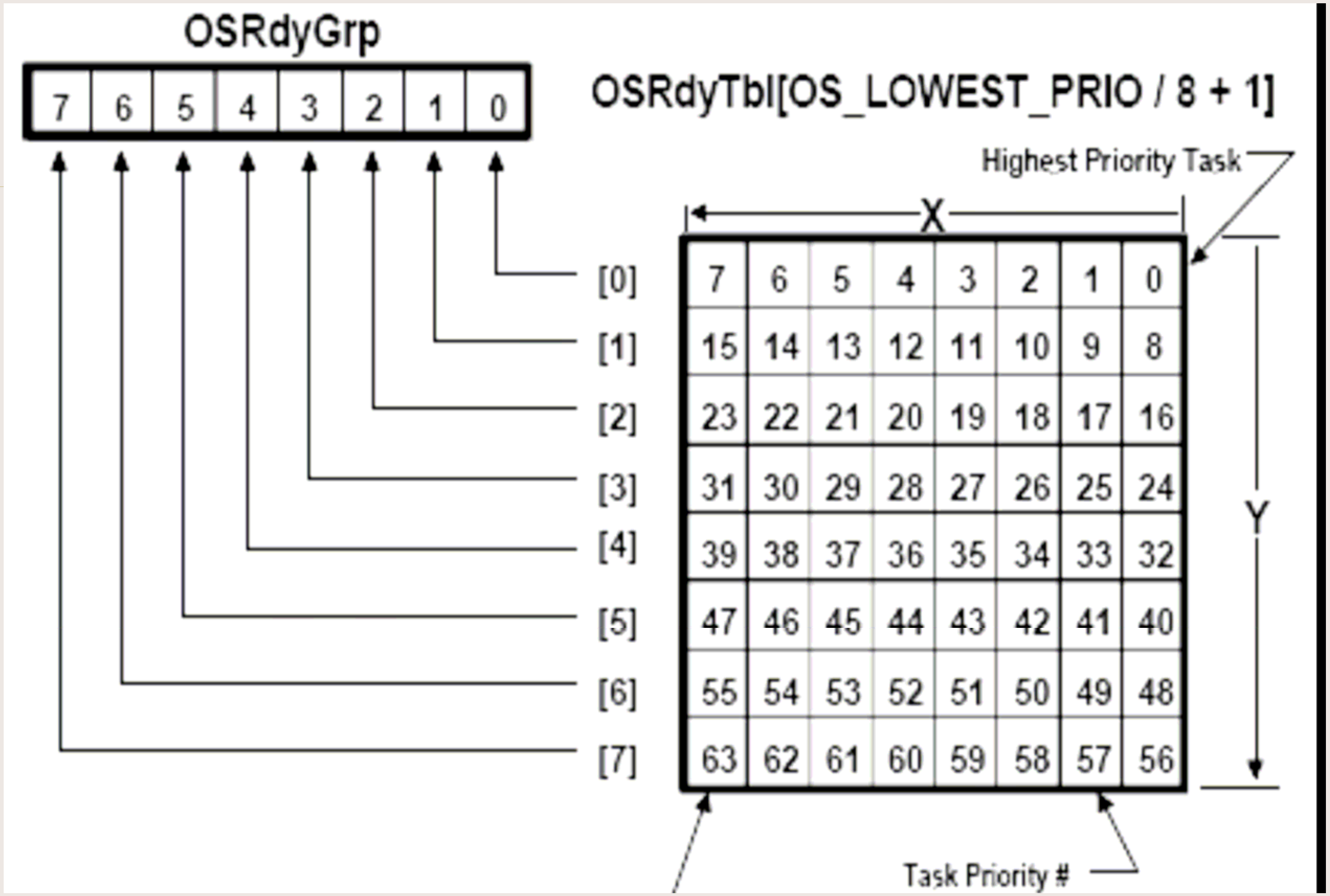
State of the task



```
typedef struct os_tcb {
OS_STK *OSTCBStkPtr;
#if OS_TASK_CREATE_EXT_EN
void *OSTCBExtPtr;
OS_STK *OSTCBStkBottom;
INT32U OSTCBStkSize;
INT16U OSTCBOpt;
INT16U OSTCBId;
#endif
struct os_tcb *OSTCBNext;
struct os_tcb *OSTCBPrev;
#if (OS_Q_EN && (OS_MAX_QS >= 2)) || OS_MBOX_EN ||
OS_SEM_EN
OS_EVENT *OSTCBEventPtr;
#endif
#if (OS_Q_EN && (OS_MAX_QS >= 2)) || OS_MBOX_EN
void *OSTCBMsg;
#endif
INT16U OSTCBDly;
INT8U OSTCBStat;
INT8U OSTCBPrio;
INT8U OSTCBX, OSTCBY, OSTCBBitX, OSTCBBitY;
#endif
} OS_TCB;
```

Task Control Block is initial in function OSTCBInit() ,this function could be used in OSTaskCreate() & OSTaskCreateEx()




OSRdyGrp
7 6 5 4 3 2 1 0
OSRdyTbl[OS_LOWEST_PRIO / 8 + 1]
Highest Priority Task
X
Y
[0] 7 6 5 4 3 2 1 0
[1] 15 14 13 12 11 10 9 8
[2] 23 22 21 20 19 18 17 16
[3] 31 30 29 28 27 26 25 24
[4] 39 38 37 36 35 34 33 32
[5] 47 46 45 44 43 42 41 40
[6] 55 54 53 52 51 50 49 48
[7] 63 62 61 60 59 58 57 56
Task Priority #

Task Scheduling

```
void OSSched (void)
{
INT8U y;
OS_ENTER_CRITICAL();
if ((OSLockNesting | OSIntNesting) == 0) {
y = OSUnMapTbl[OSRdyGrp];
OSPrioHighRdy = (INT8U)((y << 3) +
OSUnMapTbl[OSRdyTbl[y]]);
if (OSPrioHighRdy != OSPrioCur) {
OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
OSCtxSwCtr++;
OS_TASK_SW();
}
}
OS_EXIT_CRITICAL();
}
```

To Lock Scheduler

```
void OSSchedLock (void)
{
 if (OSRunning == TRUE) {
 OS_ENTER_CRITICAL();
 OSLockNesting++;
 OS_EXIT_CRITICAL();
 }
}
```

To Unlock Scheduler

```
void OSSchedUnlock (void){
   if (OSRunning == TRUE) {
      OS_ENTER_CRITICAL();
    if (OSLockNesting > 0) {
      OSLockNesting--;
      if ((OSLockNesting | OSIntNesting) == 0) {
      OS_EXIT_CRITICAL();
      OSSched();
      } else {
      OS_EXIT_CRITICAL();}
    }
    else {
      OS_EXIT_CRITICAL();
    }
   }
}
```

Interrupt Service Routine

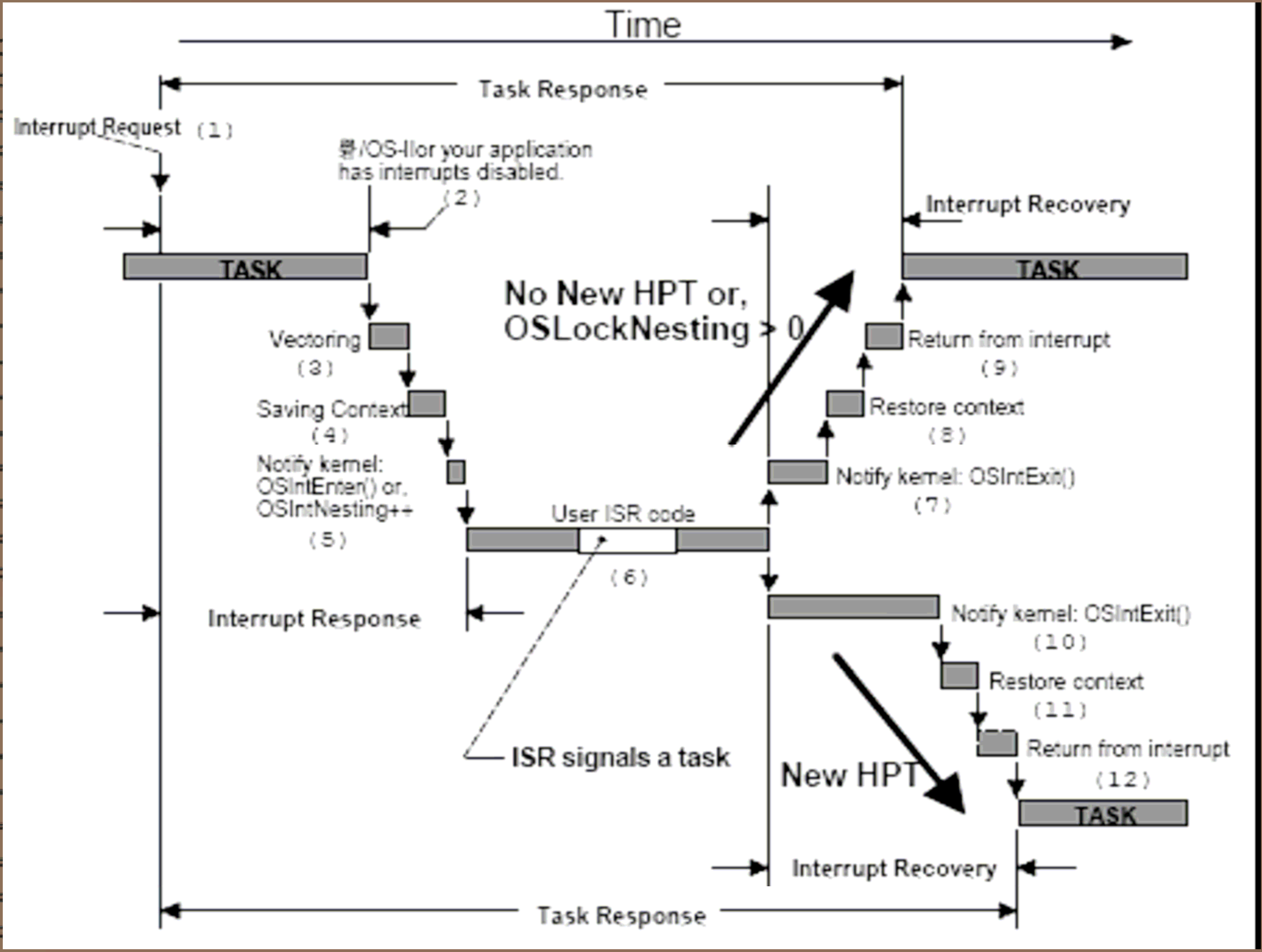
```
YourISR:
Save all CPU registers;
Call OSIntEnter() or, increment
OSIntNesting directly;
Execute user code to service ISR;
Call OSIntExit();
Restore all CPU registers;
Execute a return from interrupt
instruction;
```

Notify kernel : ISR enter

```
void OSIntEnter (void)
{
OS_ENTER_CRITICAL();
OSIntNesting++;
OS_EXIT_CRITICAL();
}
```

Notify kernel : ISR exit

```
void OSIntExit (void)
{
OS_ENTER_CRITICAL();
if ((--OSIntNesting | OSLockNesting) == 0) {
OSIntExitY = OSUnMapTbl[OSRdyGrp];
OSPrioHighRdy = (INT8U)((OSIntExitY << 3) +
OSUnMapTbl[OSRdyTbl[OSIntExitY]]);
if (OSPrioHighRdy != OSPrioCur) {
OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
OSCtxSwCtr++;
OSIntCtxSw();
}
}
OS_EXIT_CRITICAL();
}
```


Time
Task Response
Interrupt Request (1)
畀/OS-IIor your application has interrupts disabled. (2)
Interrupt Recovery
TASK
TASK
No New HPT or, OSLockNesting > 0
Vectoring (3)
Return from interrupt (9)
Saving Context (4)
Restore context (8)
Notify kernel: OSIntEnter() or, OSIntNesting++ (5)
Notify kernel: OSIntExit() (7)
User ISR code
(6)
Interrupt Response
Notify kernel: OSIntExit() (10)
Restore context (11)
ISR signals a task
Return from interrupt (12)
New HPT
TASK
Interrupt Recovery
Task Response