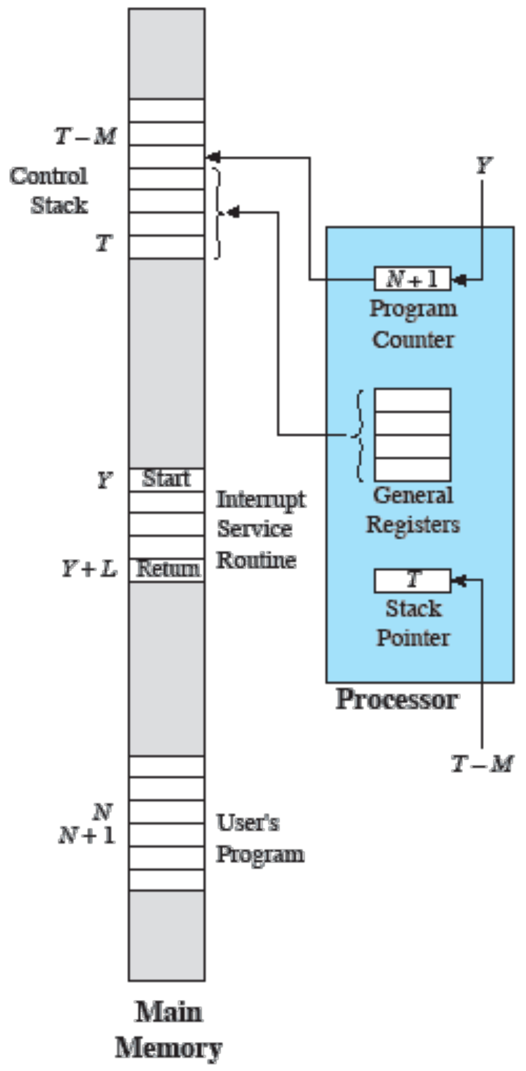
Special:
Context Switch

An Context Switch Occurs

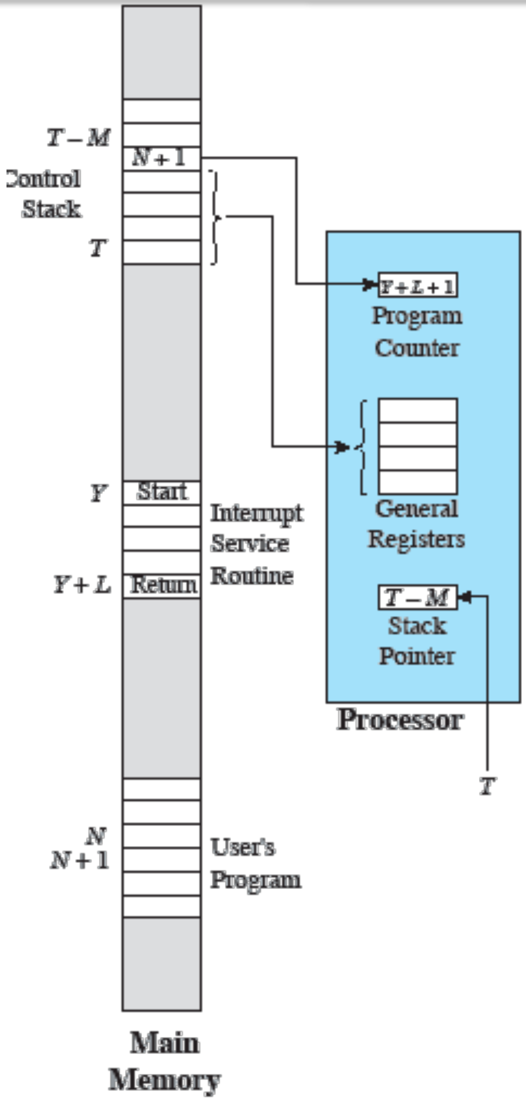
Storing a snapshot.



(a) Interrupt occurs after instruction at location N

Resuming Another Process

Restoring from a snapshot.



(b) Return from interrupt

Does it look familiar?

Yes, similar to ISRs.

PICOS18

- Kernel for Microchip PIC18 controllers
- Many function of a microkernel
  - Events
  - Interrupts
  - Counters/alarms
  - Multitasking

Schedule()

```
/*******************************************************************
 * Force a scheduler action
 *
 * @return Status    E_OK if a service is called inside an ISR
 *                   or never returns
 *******************************************************************/
StatusType Schedule(void)
{
  INTCONbits.GIEL = 0;
  kernelState |= SERVICES;
  if (kernelState & ISR)
    return (E_OK);
  kernelState &= ~SERVICES;
  if (kernelState & USER)
    SAVE_TASK_CTX(stack_low, stack_high);
  SCHEDULE;
  return (E_OK);
}
```

SAVE_TASK_CTX() – Part 1

```
#define SAVE_TASK_CTX(stack_low, stack_high)                          \
{                                                                     \
   /* Disable global interrupt. */                                    \
      _asm                                                            \
      bcf     INTCON, 6, 0                                            \
      movff   STATUS, PREINC1                                         \
      movff   WREG, PREINC1                                           \
      _endasm                                                         \
   /* Store the necessary registers to the stack. */                  \
      _asm                                                            \
      movff   BSR, PREINC1                                            \
      movff   FSR2L, PREINC1                                          \
      movff   FSR2H, PREINC1                                          \
      movff   FSR0L, PREINC1                                          \
      movff   FSR0H, PREINC1                                          \
      movff   TBLPTRU, PREINC1                                        \
      movff   TBLPTRH, PREINC1                                        \
      movff   TBLPTRL, PREINC1                                        \
      movff   TABLAT, PREINC1                                         \
      movff   PRODH, PREINC1                                          \
      movff   PRODL, PREINC1                                          \
   _endasm                                                            \
```

SAVE_TASK_CTX() – Part 2

```
/* Store the .tempdata and MATH_DATA areas. */                    \
 _asm                                                      \
    movlw   TEMP_SIZE+1                                        \
    clrf    FSR0L, 0                                     \
    clrf    FSR0H, 0                                     \
 _endasm                                                 \
    while (WREG--)                                         \
    {                                                   \
      _asm                                                 \
       movff   POSTINC0, PREINC1                                \
      _endasm                                              \
    }                                                   \
                                                        \
```

SAVE_TASK_CTX() – Part 3

```
/* Store the HW stack area. */                                  \
    _asm                                                        \
       movff   STKPTR, FSR0L                                    \
    _endasm                                                     \
       while (STKPTR > 0)                                       \
       {                                                        \
         _asm                                                   \
           movff   TOSL, PREINC1                                \
           movff   TOSH, PREINC1                                \
           movff   TOSU, PREINC1                                \
           pop                                                  \
         _endasm                                                \
       }                                                        \
```

SAVE_TASK_CTX() – Part 4

```
 /* Store the number of addresses on the HW stack */
\
   _asm                                              \
      movff   FSR0L, PREINC1                                  \
      movf    PREINC1, 1, 0                                 \
   _endasm                                              \
                                                  \
   /* Store the SW stack addr. */                              \
   _asm                                              \
      movff   stack_low, FSR0L                                \
      movff   stack_high, FSR0H                                \
      movff   FSR1L, POSTINC0                                  \
      movff   FSR1H, POSTINC0                                  \
   _endasm                                              \
}
```

_sched – Part 1

```
_sched
    GLOBAL  _sched
    #IFDEF  POSTTASKHOOK
        call    PostTaskHook
    #ENDIF
   … // skipped code here to select the next task
_restore_ctx
    GLOBAL  _restore_ctx

    movlb   0
    bsf     kernelState, 0              ; Change the kernel to USER mode
    locateTaskDescEntry
    locateStackAddrField
    loadNextAddrTo FSR0L, FSR0H         ; Extract task's stack addr
    loadNextAddrTo startAddressL, startAddressH
                                        ; Extract task's code addr
; Go chech whether the stack overflow occurred
    goto    _checkPanic
```

_sched – Part 2

```
; If the stack remains intact, restore the task's context
_restore_now
    GLOBAL  _restore_now
    movlb   0
    movff   POSTDEC1, temp
    movff   POSTDEC1, temp              ; Extract # of H/W stack entries
    clrf    STKPTR                      ; backed up previously

… // skipped a section here
```

_sched – Part 3

```
restoreNextTmpdataByte
   movff   POSTDEC1, POSTDEC0         ; Restore .tmpdata + MATH_DATA
   movf    FSR0L, w                   ; section
   btfss   STATUS, N
   bra     restoreNextTmpdataByte

   movff   POSTDEC1, PRODL            ; Restore the rest of SFRs saved
   movff   POSTDEC1, PRODH            ; in previously task swapping out
   movff   POSTDEC1, TABLAT
   movff   POSTDEC1, TBLPTRL
   movff   POSTDEC1, TBLPTRH
   movff   POSTDEC1, TBLPTRU
   movff   POSTDEC1, FSR0H
   movff   POSTDEC1, FSR0L
   movff   POSTDEC1, FSR2H
   movff   POSTDEC1, FSR2L
   movff   POSTDEC1, BSR
   movff   POSTDEC1, WREG
   movff   POSTDEC1, STATUS

   bsf     INTCON, 6                  ; Enable OS/low prior. interrupt
   retfie                             ; Exit to where TOS pointed at
```