ECE/CS 5780/6780: Embedded System Design

Scott R. Little

Lecture 11: Threads

Administrivia

- Lab 5
- 6780 Projects
Introduction to Threads

- Interrupts create a multithreaded environment with a single foreground thread (the main program), and multiple background threads (the ISRs).
- Projects where modules are loosely coupled, multiple foreground threads may be necessary.
- Threads are also referred to as lightweight processes.

Why use threads?

- Improve program responsiveness.
- Use resources efficiently.
- Decrease creation and context switching overhead.
Thread Memory

Thread States
Thread Lists

![Diagram of thread lists]

Scheduling

- Scheduling in the process by which the system determines which task to perform next.
- Scheduling decisions happen when processes or threads change state (Run → Blocked, etc.).
- Nonpreemptive scheduling is when the scheduler chooses a new thread/process after the current one terminates or blocks.
- Preemptive scheduling is when the scheduler chooses a new thread/process even though the current one is Active.
- Preemptive schedulers require increased coordination between processes/threads.
Scheduling Metrics

- CPU utilization.
- Throughput.
- Turnaround time.
- Waiting time.
- Response time.

Scheduler Types

- First-Come, First-Served.
- Shortest-Job-First.
- Priority.
- Round-Robin.
- Multi-level Queue and Variants.
A thread control block (TCB) stores information private to each thread, and it must contain:
- A pointer so that it can be chained into a linked list.
- The value of its stack pointer.
- A stack area for local variables and saved registers.

A TCB may also contain:
- Thread number, type, or name.
- Age, or how long this thread has been active.
- Priority.
- Resources that this thread has been granted.
Thread Registers

C for the Threads

```c
int Sub(int j){ int i;
    PTM = 1;  // PTM=program being executed
    i = j+1;
    return(i); }
void ProgA(){ int i;
    i=5;
    while(1) {
        PTM = 2;
        i = Sub(i); }}
void ProgB(){ int i;
    i=6;
    while(1) {
        PTM = 4;
        i = Sub(i); }
```
Thread Control Block in C

```c
struct TCB
{
    struct TCB *Next;    /* Link to Next TCB */
    unsigned char *SP;   /* Stack Pointer when idle */
    unsigned short Id;   /* output to PortT */
    unsigned char MoreStack[49]; /* more stack */
    unsigned char CCR;   /* Initial CCR */
    unsigned char RegB;  /* Initial RegB */
    unsigned char RegA;  /* Initial RegA */
    unsigned short RegX; /* Initial RegX */
    unsigned short RegY; /* Initial RegY */
    void (*PC)(void);    /* Initial PC */
};
typedef struct TCB TCBType;
typedef TCBType * TCBPtr;
```

Thread Control Block in C

```c
TCBType sys[3]=
{
    { &sys[1],   /* Pointer to Next */
      &sys[0].CCR,    /* Initial SP */
      1,        /* Id */
      0,         /* CCR,B,A,X,Y */
      ProgA }, /* Initial PC */
    { &sys[2],   /* Pointer to Next */
      &sys[1].CCR,    /* Initial SP */
      2,        /* Id */
      0,         /* CCR,B,A,X,Y */
      ProgA }, /* Initial PC */
    { &sys[0],   /* Pointer to Next */
      &sys[2].CCR,    /* Initial SP */
      4,        /* Id */
      0,         /* CCR,B,A,X,Y */
      ProgB } }; /* Initial PC */
```
Preemptive Thread Scheduler in C

```c
TCBPtr RunPt; /* Pointer to current thread */
void main(void){
    DDRT = 0xFF; /* Output running thread on Port T */
    DDRM = 0xFF; /* Output running program on Port M */
    RunPt = &sys[0]; /* Specify first thread */
    asm sei
    TFLG1 = 0x20; /* Clear C5F */
    TIE = 0x20; /* Arm C5F */
    TSCR1 = 0x80; /* Enable TCNT*/
    TSCR2 = 0x01; /* 2MHz TCNT */
    TIOS |= 0x20; /* Output compare */
    TC5 = TCNT+20000;
    PTT = RunPt->Id;
    asm ldx RunPt
    asm lds 2,x
    asm cli
    asm rti
} /* Launch First Thread */
```

Preemptive Thread Scheduler in C (cont)

```c
void interrupt 13 ThreadSwitch(){
    asm ldx RunPt
    asm sts 2,x
    RunPt = RunPt->Next;
    PTT = RunPt->Id; /* PortH=active thread */
    asm ldx RunPt
    asm lds 2,x
    TC5 = TCNT+20000; /* Thread runs for 10 ms */
    TFLG1 = 0x20; } /* ack by clearing C5F */
```
void create(void (*program)(void), int TheId) {
    TCBPtr NewPt;    // pointer to new thread control block
    NewPt = (TCBPtr)malloc(sizeof(TCBType));  // new TCB
    if (NewPt == NULL) return;
    NewPt->SP = &(NewPt->CCR);    /* Stack Pointer when not running */
    NewPt->Id = TheId;            /* Visualize active thread */
    NewPt->CCR = 0x40;            /* Initial CCR, I=0 */
    NewPt->RegB = 0;              /* Initial RegB */
    NewPt->RegA = 0;              /* Initial RegA */
    NewPt->RegX = 0;              /* Initial RegX */
    NewPt->RegY = 0;              /* Initial RegY */
    NewPt->PC = program;          /* Initial PC */
    if (RunPt) {
        NewPt->Next = RunPt->Next;
        RunPt->Next = NewPt;       /* will run Next */
    } else
        RunPt = NewPt;            /* the first and only thread */
}

Scott R. Little (Lecture 11: Threads) | ECE/CS 5780/6780 | 19 / 19