EECE 276
Embedded Systems

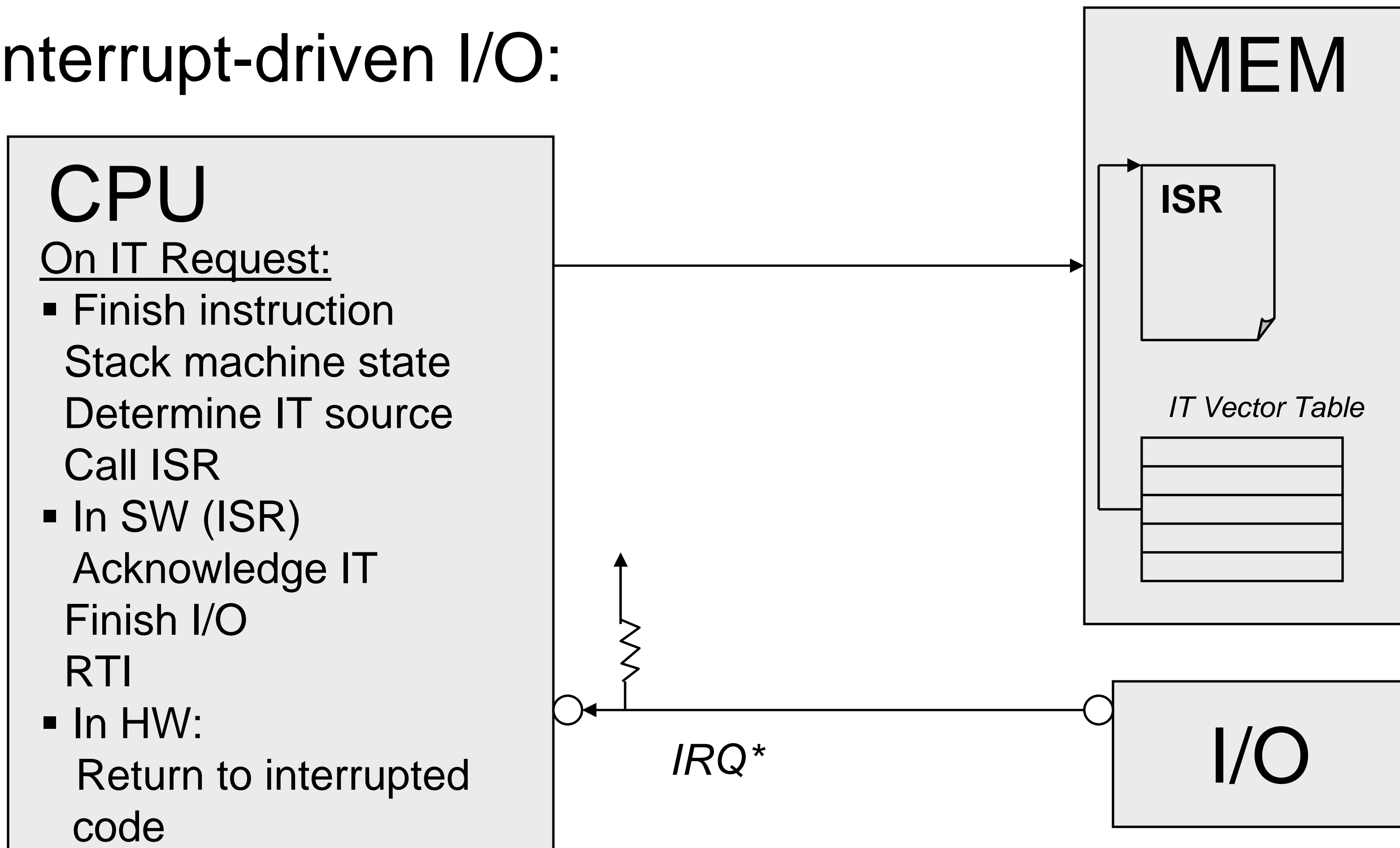
Interrupts: Basics

Shared data problem

Latency

Recap: IT Technique

Interrupt-driven I/O:



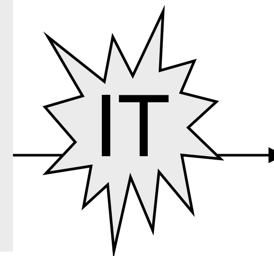
IT Technique: Key points

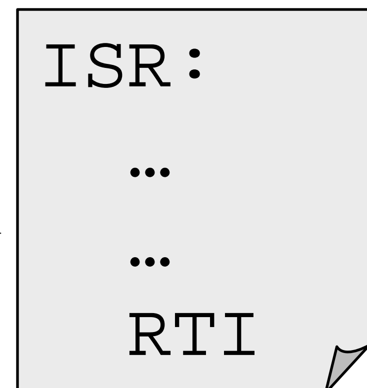
- IT is always generated by a special event
  - HW (mostly), SW (sometime)
- If enabled, an IT can be serviced between **any** two instructions
  - Instructions are always *fully* executed

NOTE: **C statements often mean multiple instructions**

```
void foo(int i) {
   int j;
   j = i + 123;
   …
}
```

```
LDAA $8,X
ADDA #123
STAA $6,X
```





IT Technique: Key points

- All registers must be saved/restored at the beginning/end of the ISR
  - Saving/restoring the *context* (a.k.a. *context switch*)
- Locating the ISR: IT vector table
- Multiple, simultaneous IT-s: priority scheme
  - HC11/12: fixed scheme on internals, single external source
  - Others: IT priority controller (extra HW)
- Interrupting the ISR: IT nesting
  - HC11/12: Upon entry to an ISR, IT-s are disabled
  - Others: Higher priority IT-s are enabled

Shared Data Problem

- Code monitors two temperatures
- Temperatures must be equal
- If not -> ALARM
- vReadTemps() is the ISR, called periodically

Behavior:

The code occasionally sets off the ALARM, even if everything seems normal. ???

```
static int iTemps[2];
void interrupt vReadTemps() {
  iTemps[0] = // read in value from HW;
  iTemps[1] = // read in value from HW:
}
void main(void) {
  int iTemp0, iTemp1;
  while (TRUE) {
    iTemp0 = iTemps[0];
    iTemp1 = iTemps[1];
    if (iTemp0 != iTemp1) {
      // Set off ALARM
    }
  }
}
```

Shared Data Problem

Sequence of events:

1. main: iTemp0 = iTemps[0];
2. ISR: updates iTemps[0] and iTemps[1]
3. main: iTemp1 = iTemps[1];
4. main: iTemp0 != iTemp1 → ALARM!

Alternative main(): →

Does it fix the problem? **NO!**

IT always comes at the wrong time.

```
….
void main(void) {
while (TRUE) {
    if (iTemps[0] != iTemps[1]) {
      // Set off ALARM
    }
  }
```

Shared Data Problem

Source of the problem:

iTemps[] array is shared between the main() and the ISR. If IT happens while main() is using the array -> the data may be in an inconsistent state.

Solving the problem:

Enable/disable ITs

Atomic/critical section →

```
…
void main(void) {
 int iTemp0, iTemp1;
 while (TRUE) {
   disableIT();
   iTemp0 = iTemps[0];
   iTemp1 = iTemps[1];
   enableIT();
   if (iTemp0 != iTemp1) {
     // Set off ALARM
   }
 …
```

Interrupt latency

How fast will a system react to interrupts? Depends on:

1. Max. time while IT-s are disabled.
2. Max. time taken to execute higher priority IT-s.
3. Time taken by ISR invocation (context save, etc.) and return (context restore)
4. “Work” time in ISR to generate a response.

Values:

For 3: see processor docs.

Others: count instructions – does not work well for processors with cache!

General rule: WRITE SHORT IT SERVICE ROUTINES!

Interrupt latency: Disabling Interrupts

Example system:

- Must disable IT-s for 125uS to process pressure variables.
- Must disable IT-s for 250uS to manage timer
- Must respond to a network IT within 600uS, the network ISR takes 300uS to execute.

Will it work?

Interrupt latency: Disabling Interrupts

Will it work? Yes, as the longest time IT-s are disabled is 250uS and 300uS is required for the network ISR. WCRT = 550uS < 600 uS.

Will it work with a processor of half the speed?

No!

WCRT = 500uS + 600uS = 1100uS > 600uS

Alternative to disabling IT-s

```
int iTempAs[2];
int iTempBs[2];
bool fUsingB = FALSE;
void interrupt vReadTemps() {
    if(fUsingB) {
        iTempAs[0] = // read from HW
        iTempAs[1] = // read from HW
    } else {
        iTempBs[0] = // read from HW
        iTempBs[1] = // read from HW
    }
}
```

Two sets of variables

One flag to control which set is used

Alternative to disabling IT-s

```
void main () {
    while (TRUE) {
        if(fUsingB) {
            if (iTempBs[0] != iTempBs[1]) {
                // set off ALARM
            }
        } else {
            if (iTempAs[0] =! iTempAs[1]) {
                // set off ALARM
            }
        }
        fUsingB = !fUsingB;
    }
}
```

Assumption:

Changing the flag is an atomic operation!