PS3 Programming

Week 4. Events, Signals, Mailbox

Chap 7 and Chap 13

Outline

- Event
  - PPU’s event
  - SPU’s event
- Mailbox
- Signal
- Homework

EVENT

PPU’s event

- PPU can enable events when creating SPE’s context by `spe_context_create`
  - Set the flag SPE_EVENTS_ENABLE
- Three steps to create an event
  1. Create an event handler
  2. Create and initialize the event
  3. Register the event with the event handler

Event handler

- The definition of spe event handler

```
typedef struct spe_event_unit {
   spe_context_ptr_t spe;
   unsigned int events;
   spe_event_data_t data;
} spe-event_unit_t
```

- There are 5 (SPE) events
  - Ex: SPE_EVENT_SPE_STOPPED
- Events are listened/waited by calling spe_event_wait

Example: monitor SPE stop

```
#include <stdio.h>
#include <stdlib.h>
#include <libspe2.h>
#define MAX_EVENTS 16
extern spe_program_handle_t spu_events;
int main(int argc, char **argv) {
   int i, event_count;
   spe_context_ptr_t ctx;      /* Context */
   unsigned int entry_point;   /* Start address */
   int retval;                 /* Return value */
   spe_stop_info_t stop_info;  /* Stop info */
   spe_event_unit_t events[MAX_EVENTS]; /* Events to be
                                           received */
   /* Create the context */
   ctx = spe_context_create(SPE_EVENTS_ENABLE, NULL);
```

Continue

```
spe_event_handler_ptr_t ehandler;  /* Event handler */
spe_event_unit_t event;      /* Event to be handled */

/* Create an event handler and register event */
ehandler = spe_event_handler_create();
event.spe = ctx;
event.events = SPE_EVENT_SPE_STOPPED;
spe_event_handler_register(ehandler, &event);
 .../* Load the program handle into the context */
 .../* Execute the program inside the context */
 /* Receive events and analyze stop information */
event_count = spe_event_wait(ehandler, events,
                             MAX_EVENTS, 10);
printf("Number of events detected: %d\n", event_count);
.../* process the event */
```

SPU’s Event

- SPU has 12 (MFC) events
  - – Ex: MFC_TAG_STATUS_UPDATE
- Three steps for event handling
  1. Select events of interest
  2. Recognize event occurring
     - Waiting, polling, and interrupt
  3. Acknowledge events

ISR (interrupt service routine)

- Located at a special address (0x0000)
  - The `.interrupt` section of SPU ELF file
  - Register the route by declaring the ISR at the .interrupt section

```
void interrupt_service(void)
__attribute__ ((section (".interrupt")));
```

- Use spu_ienable() and spu_idisable() to start and stop the interrupt signal

Example

Measure time in SPU. See 11.3

```
void interrupt_service(void) {

    int dec = spu_read_decrementer();
    printf("ISR: Decrementer = %d.\n", dec);

    /* End loop in main function */
    check_value = 1;

    /* Acknowledge event detection */
    spu_write_event_ack(MFC_DECREMENTER_EVENT);

    /* Return to main function */
    asm("iret");
}
```

Main function

```c
#include <spu_mfcio.h>
void interrupt_service(void)
   __attribute__ ((section (".interrupt")));
volatile unsigned int check_value = 0;

int main(unsigned long long speid, unsigned long long
argp, unsigned long long envp) {
   /* Enable interrupt processing */
   spu_ienable();
   /* Write to the event mask */
   spu_write_event_mask(MFC_DECREMENTER_EVENT);
   /* Write to the decrementer and begin countdown */
   spu_write_decrementer(10000);
   /* Loop while waiting for interrupt */
   while(check_value == 0);
   return 0;
}
```

MAILBOX

Mailbox

- Small message (4 bytes) between SPU/PPU
- SPU to PPU
  - SPU writes to out mbox
  - PPU reads SPU’s out mbox
- PPU to SPU
  - PPU write to SPU’s in mbox
  - SPU reads its in mbox
- Use event to notify the arrival of mails

SPU’s code

```
include <spu_mfcio.h>
void interrupt_service(void)
   __attribute__ ((section (".interrupt")));
volatile unsigned int check_value = 0;
int main(unsigned long long speid, unsigned long
              long argp, unsigned long long envp) {
   unsigned int mbox_content;
   /* Write to the event mask */
   spu_write_event_mask(MFC_IN_MBOX_AVAILABLE_EVENT);
   /* Enable interrupt and wait for the interrupt */
   spu_ienable();
   while(!check_value);
   /* Read mailbox and display result */
   mbox_content = spu_read_in_mbox();
   return 0;
}
```

SPU’s ISR

```
void interrupt_service(void) {
   spu_write_event_ack(MFC_IN_MBOX_AVAILABLE_EVENT);
   check_value++;
   asm("iret");
}
```

PPU’s code

```
int main() {
   int retval; spe_context_ptr_t spe;
   unsigned int mbox_data[1];
   /* Create context, load program, create thread*/
   ...
   /* Write a value to the SPE's Incoming Mailbox */
   mbox_data[0] = 0x12345678;
   if(spe_in_mbox_status(spe))
      spe_in_mbox_write(spe, mbox_data, 1,
                        SPE_MBOX_ALL_BLOCKING);
   printf("Sent data = %x\n", mbox_data[0]);

   /* Wait thread finish and deallocate the context */
   ...
}
```

SIGNALS

Signal and mailbox

- Same
  - 32 bits, controlled by MFC, used for control
- Differences
  - Signal has tag group id (DMA's tag)
  - Signal can be used directly between SPUs
  - Signal has 1-1, n-1; mailbox only has 1-1
  - Signal has two channels; mailbox has in/out mbox
  - Signal will not be removed after read

Signal

- The signal send (SPU) is similar to mfc_put

```
mfc_sndsig(volatile void *ls, uint64_t ea,
uint32_t tag, uint32_t, tid, uinit322_t rid)
```

  - ls : source address at local store
  - ea: distination
  - tag, tid, rid are the same to mfc_put
- We will skip the details of using it since it is too complicated

HOMEWORK

Assignments

- Run the examples in chapter 13
- Continue the Huffman coding project.
  - Use mailbox/event/ISR to inform the address of next block to be process
  - You can also try to use one of the SPU as the master.