Lecture 25: Asynchronous Hello Cell

Prof. Aaron Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology

hello_be1 – async version – spu code

Makefile

```
Makefile
PROGRAM_spu . = hello_spu
LIBRARY_spu . = hello_spu.a
ECPPFLAGS gcc . = -s
SPU_TIMING . = 1
ACC_OPT_LEVEL . = -04
include $(TOP)/buildutils/make.footer

hello_spu.c (for the hello_be1 example)
#include <stdio.h>
int main(unsigned long long spid,
         unsigned long long arge,
         unsigned long long envp)
{
    printf("Hello World\n");
    return 0;
}
```

hello_be1 – async version – ppu code (1)

Makefile

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <libspe2.h>
#include <pthread.h>
```

hello_be1 – async version – ppu code (2)

```
typedef struct ppu_thread_data{
    spe_context_ptr_t context;
    pthread_t pthread;
    unsigned int entry;
    unsigned int flags;
    void *argp;
    void *envp;
    spe_stop_info_t stopinfo;
} ppu_thread_data_t;
```
hello_be1 – async version – ppu code (3)

```c
void *ppu_pthread_function(void *arg)
{
    ppu_pthread_data_t *datap = (ppu_pthread_data_t *)arg;
    int rc;
    rc = spe_context_run(datap->context, &datap->entry,
                         datap->flags, datap->argc,
                         datap->argv, datap->envp, datap->stopinfo);
    pthread_exit(NULL);
}
```

hello_be1 – async version – ppu code (4)

```c
extern spe_program_handle_t hello_spu;

int main(void)
{
    ppu_pthread_data_t data;
    data.context = spe_context_create(0, NULL);
    spe_program_load(data.context, &hello_spu);
    data.entry = SPE_DEFAULT_ENTRY;
    data.flags = 0;
    data.argv = NULL;
    data.envp = NULL;
    pthread_create(&data.pthread, NULL,
                   &ppu_pthread_function, &data);
    pthread_join(data.pthread, NULL);
    spe_context_destroy(data.context);
    return 0;
}
```

simple-multispu/simple – spu code

**Makefile**

```make
PROGRAMS_spu := simple_spu
LIBRARY_membed := lib_simple_spu.a
include $(CELL_TOP)/buildutils/make.footer

simple_spu.c
```

```c
#include <stdio.h>

int main(unsigned long long id) {
    printf("Hello Cell (%llx)\n", id);
    return 0;
}
```

**Makefile**

```make
DIRS_spu := simple
PROGRAM_ppu := simple
IMPORTS_spu := spu/lib_simple_spu.a -lspe2 -lpthread
INSTALL_DIR = $(SDKBIN)/samples
INSTALL_FILES = $(PROGRAM_ppu)
include $(CELL_TOP)/buildutils/make.footer

simple.c
```

```c
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <libspe2.h>
#include <pthread.h>
```
simple-multispu/simple – ppu code (2)

```c
extern spe_program_handle_t simple_spu;
#define SPU_THREADS 6

void *ppu_thread_function(void *arg) {
    spe_context_ptr_t ctx;
    unsigned int entry = SPE_DEFAULT_ENTRY;
    ctx = *((spe_context_ptr_t *)arg);
    if (spe_context_run(ctx, entry, 0, NULL, NULL, NULL) < 0) {
        perror("Failed running context");
        exit (1);
    }
    pthread_exit(NULL);
}

int main() {
    int i;
    spe_context_ptr_t ctxs[SPU_THREADS];
    pthread_t threads[SPU_THREADS];
    ...}
```

simple-multispu/simple – ppu code (3)

```c
/* Create several SPE threads to execute 'simple_spu'. */
for(i=0; i<SPU_THREADS; i++) {
    /* Create context */
    if ((ctxs[i] = spe_context_create (0, NULL)) == NULL) {
        perror("Failed creating context");
        exit (1);
    }
    /* Load program into context */
    if (spe_program_load (ctxs[i], &simple_spu)) {
        perror("Failed loading program");
        exit (1);
    }
    /* Create thread for each SPE context */
    if (pthread_create (&threads[i], NULL, &ppu_thread_function, &ctxs[i])) {
        perror("Failed creating thread");
        exit (1);
    }
```

simple-multispu/simple – ppu code (4)

```c
/* Wait for SPE-thread to complete execution. */
for (i=0; i<SPU_THREADS; i++) {
    if (pthread_join (threads[i], NULL)) {
        perror("Failed pthread join");
        exit (1);
    }
}
printf("\nThe program has successfully executed.\n");
return (0);
```

simple-multispu/simple - let’s run it!

```
I think I have a correct solution. I'll test it now.
```

```
I think I have a correct solution. I'll test it now.
```

```
Hello! I just joined a new team. I'm excited to see how this goes.
```

```
Hello! I just joined a new team. I'm excited to see how this goes.
```

```
I'm a bit nervous about this new project. I hope I can handle it.
```

```
I'm a bit nervous about this new project. I hope I can handle it.
```

```
The program has successfully executed.
```

```
The program has successfully executed.
```

```
```