Compiling Program

A regular C-program

For this exercise, it is important that you have already compiled a core – either *pearl*, or *pearl_is3.* We will use a simple example to get started.

- Login to one of the servers mentioned in the startup guide.
- Change directory to ~/hive/examples/add/orig and check the files in the folder as follows

```
-bash-3.00$ cd ~/hive/examples/add/orig/
-bash-3.00$ pwd
/home/pd/pd0801/hive/examples/add/orig
-bash-3.00$ ll
total 16
-rw-r--r--  1 pd0801 pd 113 Oct 22 13:02 add_c.c
-rw-r--r--  1 pd0801 pd  84 Oct 22 13:01 add_c.h
-rw-r--r--  1 pd0801 pd 222 Oct 22 13:12 host.c
-rw-r--r--  1 pd0801 pd 214 Oct 22 13:14 Makefile
```

- You can inspect the files using a text editor like in the earlier exercise (e.g. vim or nedit). Following is a short description of each file.
  - o add_c.c: defines a function add that adds and multiply two numbers.
  - o add_c.h: declares the function add in the earlier file.
  - o host.c: calls the function add.
  - o Makefile: defines the compilation flow. Note the target that
- Compile the program by using the following command (gmake crun). You will see the output as follows

```
-bash-3.00$ gmake crun
/mnt/o4/home/pd/pd0801/hive/examples/add/orig/host.c:4: warning: unused parameter
‘argc’
/mnt/o4/home/pd/pd0801/hive/examples/add/orig/host.c:4: warning: unused parameter
‘argv’
Add output received: 7
Multiply output received: 12
Expected add output: 7
Expect multiply output:12
/mnt/o4/home/pd/pd0801/hive/examples/add/orig/crun Succeeded
=================
Report:
/mnt/o4/home/pd/pd0801/hive/examples/add/orig/crun Succeeded
=================
```

- You should see that the compilation was successful, and that the numbers received from the add function are also correct.

Compiling C-program for custom processors

- We shall now modify the regular c-program to make it run on the custom processors generated earlier.
- Make a new folder in **~/hive/examples/add/** called **pearl** as shown below, and copy/rename some files as follows:

```
-bash-3.00$ pwd
/home/pd/pd0801/hive/examples/add
-bash-3.00$ cp -R orig pearl
-bash-3.00$ cd pearl
-bash-3.00$ ll
total 16
-rw-r--r--  1 pd0801 pd 113 Oct 22 13:02 add_c.c
-rw-r--r--  1 pd0801 pd  84 Oct 22 13:01 add_c.h
-rw-r--r--  1 pd0801 pd 222 Oct 22 13:12 host.c
-rw-r--r--  1 pd0801 pd 214 Oct 22 13:14 Makefile

-bash-3.00$ cp add_c.c add.hive.c
-bash-3.00$ cp add_c.h add.hive.h
-bash-3.00$ ll
total 24
-rw-r--r--  1 pd0801 pd 178 Oct 22 15:40 add_c.c
-rw-r--r--  1 pd0801 pd  90 Oct 22 15:40 add_c.h
-rw-r--r--  1 pd0801 pd 178 Oct 21 14:30 add.hive.c
-rw-r--r--  1 pd0801 pd  90 Oct 16 14:11 add.hive.h
-rw-r--r--  1 pd0801 pd 841 Oct 21 13:30 host.c
-rw-r--r--  1 pd0801 pd 455 Oct 21 14:39 Makefile
```

Make the following changes

- **Makefile**
  - Add another method **sched** in line 3. It should like below.
    ```
    METHODS = crun sched
    ```
  - Change the file to be used as input in line 9. It should like below
    ```
    add_FILES    = add.hive.c
    ```
  - Add the following lines to the file.
    ```
    CUSTOM_HOST = hive
    hive_CC     = $(HIVEBIN)/hivecc -m pearl -mdir $(HIVECORES)
    hive_CPP = cpp -D__HIVECC
    ```
- **host.c**
  - Replace the first two lines by following lines. These are needed to use the *pearl* processor we have generated earlier and to use silicon hive run-time libraries. Further, since we are using two methods in this part (crun and sched), we have to use appropriate header files.

```
#include <pearl_system.h>
#define CELL pearl

#include <hrt/embed.h>

#ifdef C_RUN
#include "add.hive.h"
#else
#include "add.map.h"
#ifndef HRT_CRUN_LS
```

```
#include "add.hive.stubs.h"
#endif
#endif
```

  - In the original version, we called the function add directly with parameters. When the program is executing on the processor, we first load the program to be executed, then the initial variables, start the processor, and then read the results. Thus, the original call of **add** is replaced by the following lines:

```
//Loading the program
hrt_cell_load_program_id(CELL, add);

//Loading the initial variables
hrt_indexed_store(CELL, int, add_inputs, 0, inputs[0]);
hrt_indexed_store(CELL, int, add_inputs, 1, inputs[1]);

//Starting the program
add();

//Reading back the results
add_output = hrt_scalar_load(CELL, int, add_output_gen);
mul_output = hrt_scalar_load(CELL, int, mul_output_gen);
```

  - Add the following lines to produce an error if the results are wrong (just before the last return statement)

```
  if (add_output != expected_add_output || mul_output != expected_mul_output) {
    hrt_error("Wrong output received");
    return 1;
  }
```

- add.hive.c
  - Change the included header file from "**add_c.h**" to "**add.hive.h**", and add a new header file **<hive/support.h>**. Your include definitions should look like this.

    ```
    #include <hive/support.h>
    #include "add.hive.h"
    ```

  - Since the input and output data are now stored in memory and not passed in the function call, they have to be explicitly declared as follows.

    ```
    int add_inputs[2];
    int add_output_gen, mul_output_gen;
    ```

  - Change the function declaration to void and the keyword ENTRY – this tells the processor to start from this function. Further, remove the pointers from add_output, and mul_output. The function should now look like this.

    ```
    void add(void) ENTRY
    {
      add_output_gen = add_inputs[0] + add_inputs[1];
      mul_output_gen = add_inputs[0] * add_inputs[1];
    }
    ```

- **add.hive.h**
  - Change the variable **_add_c_h** to **_add_hive_h** in the whole file.
  - Add the declaration of input and output variables used in **add.hive.c**.
  - Change the definition of **add** function.
  - The file should now look like this

    ```
    #ifndef _add_hive_h
    #define _add_hive_h
    ```

```
extern int add_inputs[2];
extern int add_output_gen, mul_output_gen;

extern void add(void);
#endif /* _add_hive_h */
```

- We are now ready to compile the program. First compile for **crun** and check if everything is ok. The output and the directory structure should look like this.

```
-bash-3.00$ gmake clean crun
/mnt/o4/home/pd/pd0801/hive/examples/add/v1/crun Succeeded
-bash-3.00$ ll
total 32
-rw-r--r--  1 pd0801 pd  178 Oct 22 15:40 add_c.c
-rw-r--r--  1 pd0801 pd   87 Oct 22 16:27 add_c.h
-rw-r--r--  1 pd0801 pd  231 Oct 22 16:48 add.hive.c
-rw-r--r--  1 pd0801 pd  160 Oct 22 16:48 add.hive.h
drwxr-xr-x  2 pd0801 pd 4096 Oct 22 16:48 crun
drwxr-xr-x  3 pd0801 pd 4096 Oct 22 16:48 hivecc_intermediates
-rw-r--r--  1 pd0801 pd 1119 Oct 22 16:47 host.c
-rw-r--r--  1 pd0801 pd  329 Oct 22 15:52 Makefile
```

- Let us now compile the program on our pearl processor by doing a **sched** run. You will see a lot of output and two new folders being created as follows.

```
-bash-3.00$ gmake sched
.....
.....
/mnt/o4/home/pd/pd0801/hive/examples/add/v1/sched Succeeded
-bash-3.00$ ll
total 40
-rw-r--r--  1 pd0801 pd  178 Oct 22 15:40 add_c.c
-rw-r--r--  1 pd0801 pd   87 Oct 22 16:27 add_c.h
-rw-r--r--  1 pd0801 pd  231 Oct 22 16:48 add.hive.c
-rw-r--r--  1 pd0801 pd  160 Oct 22 16:48 add.hive.h
drwxr-xr-x  2 pd0801 pd 4096 Oct 22 16:48 crun
drwxr-xr-x  3 pd0801 pd 4096 Oct 22 16:48 hivecc_intermediates
-rw-r--r--  1 pd0801 pd 1119 Oct 22 16:47 host.c
drwxr-xr-x  3 pd0801 pd 4096 Oct 22 16:52 html
-rw-r--r--  1 pd0801 pd  329 Oct 22 15:52 Makefile
drwxr-xr-x  2 pd0801 pd 4096 Oct 22 16:52 sched
```

- If you check in the folder **sched**, you will see a file called **statistics.txt** that gives a summary of the number of clock cycles taken to execute the program etc.

```
-bash-3.00$ cat sched/ statistics.txt
```

- The **html** folder has an **add_add.hive.html** file that can be opened using **firefox** and various properties can be inspected.

```
-bash-3.00$ cd html
-bash-3.00$ firefox add_add.hive.html &
```

Questions

1) What is the total number of clock cycles to execute the program on **pearl** processor?
2) How many operations are executed in total?

Using a different processor

- Let us now use the modified processor with 3 issue slots for mapping the same application and see if it makes a difference. Create a new folder in **~/hive/examples/add** as follows

```
-bash-3.00$ cd ~/hive/examples/add
-bash-3.00$ cp -R pearl pearl_is3
-bash-3.00$ cd pearl_is3/
```

- Change all occurrences of **pearl** with **pearl_is3** in Makefile and host.c. Let us recompile the program now

```
-bash-3.00$ gmake clean crun sched
/mnt/o4/home/pd/pd0801/hive/examples/add/pearl_is3/crun Succeeded
.....
.....
/mnt/o4/home/pd/pd0801/hive/examples/add/pearl_is3/sched Succeeded
```

Questions

3) What is the total number of clock cycles to execute the program on **pearl_is3** processor?
4) How many operations are executed in total on **pearl_is3**?