Hands-on – The Hello World! Program
Course Objectives

- You will learn how to write, build and run “Hello World!” on the Cell System Simulator
- Navigate through the basic build process and make files
- Familiarize with gcc and xlc compilers
- There are three different versions of “Hello World!” used in this session
  - PPE only,
  - SPE only, and
  - Cell BE, i.e. using both PPE and SPE

**Trademarks** - Cell Broadband Engine™ is a trademark of Sony Computer Entertainment, Inc.
How to get “Hello World!”

- **Pre-requisites**
  - Toolchain
  - Compiler

- **Build Process**

- **Source Code**
  - Makefiles
  - Source PPE
  - Source SPE

- **Simulator**
  - Getting the binary into the simulator
  - Running the binary
Three Different Versions of “Hello World!”

- PPE only
- SPE only
- Synergistic PPE and SPE
  - Synchronous
  - Asynchronous (to be introduced later)
“Hello World!” – PPE Only

- **PPU program**
  - just like any “Hello World!” program one would write

```c
#include <stdio.h>

int main(void)
{
    printf("Hello world!\n");
    return 0;
}
```

- **Makefile**
  - `make.footer` included to set up compiler and compiler flags
  - `PROGRAM_ppu` tells make to use PPC cross-compilation

```
PROGRAM_ppu = hello_ppu
include $(CELL_TOP)/make.footer
```

**PROGRAM_ppu** tells make to use PPC compiler
“Hello World!” – SPE Only

- **SPU Program**

```c
#include <stdio.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

- **SPU Makefile**

```
PROGRAM_spu := hello_spu
include $(CELL_TOP)/make.footer
```

**PROGRAM_spu tells make to use SPE compiler**
Synergistic PPE and SPE (SPE Embedded)

- Applications use software constructs called SPE contexts to manage and control SPEs.
- Linux schedules SPE contexts from all running applications onto the physical SPE resources in the system for execution according to the scheduling priorities and policies associated with the runnable SPE contexts.
- libspe provides the means for communication and data transfer between PPE threads and SPEs.
How a PPE program embeds an SPE program?

- 4 basic steps must be done by the PPE program
  - Create an SPE context.
  - Load an SPE executable object into the SPE context local store.
  - Run the SPE context. This transfers control to the operating system, which requests the actual scheduling of the context onto a physical SPE in the system.
  - Destroy the SPE context.
SPE context creation

- **spe_context_create** - Create and initialize a new SPE context data structure.

  ```c
  #include <libspe2.h>
  spe_context_ptr_t spe_context_create(unsigned int flags,
                                       spe_gang_context_ptr_t gang)
  ```

  - **flags** - A bit-wise OR of modifiers that are applied when the SPE context is created.
  - **gang** - Associate the new SPE context with this gang context. If NULL is specified, the new SPE context is not associated with any gang.
  - On success, a pointer to the newly created SPE context is returned.
**spe_program_load**

- **spe_program_load** - Load an SPE main program.

  ```c
  #include <libspe2.h>

  int spe_program_load (spe_context_ptr_t spe,
                        spe_program_handle_t *program)
  ```

  - `spe` - A valid pointer to the SPE context for which an SPE program should be loaded.
  - `program` - A valid address of a mapped SPE program.
spe_context_run

- **spe_context_run** - Request execution of an SPE context.

```c
#include <libspe2.h>

int spe_context_run(spe_context_ptr_t spe, unsigned int *entry, unsigned int runflags, void *argp, void *envp, spe_stop_info_t *stopinfo)
```

- `spe` - A pointer to the SPE context that should be run.
- `entry` - Input: The entry point, that is, the initial value of the SPU instruction pointer, at which the SPE program should start executing. If the value of entry is SPE_DEFAULT_ENTRY, the entry point for the SPU main program is obtained from the loaded SPE image. This is usually the local store address of the initialization function `crt0`.
- `runflags` - A bit mask that can be used to request certain specific behavior for the execution of the SPE context. 0 indicates default behavior.
- `argp` - An (optional) pointer to application specific data, and is passed as the second parameter to the SPE program.
- `envp` - An (optional) pointer to environment specific data, and is passed as the third parameter to the SPE program.
- `stopinfo` - An (optional) pointer to a structure of type `spe_stop_info_t`
spe_context_destroy

- **spe_context_destroy** - Destroy the specified SPE context.

  ```
  #include <libspe2.h>
  int spe_context_destroy (spe_context_ptr_t spe)
  ```

- **spe** - Specifies the SPE context to be destroyed
- On success, 0 (zero) is returned, else -1 is returned
“Hello World!” – Synergistic PPE and SPE (SPE Embedded)

- **SPU program**
  - Same as for SPE only

- **SPU Makefile**

```makefile
PROGRAM_spu := hello_spu
LIBRARY_embed := hello_spu.a
include $(CELL_TOP)/make.footer
```
“Hello World!” – PPU program

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

extern spe_program_handle_t hello_spu;

int main(void)
{
    spe_context_ptr_t speid;
    unsigned int flags = 0;
    unsigned int entry = SPE_DEFAULT_ENTRY;
    void * argp = NULL;
    void * envp = NULL;
    spe_stop_info_t stop_info;
    int rc;

    // Create an SPE context
    speid = spe_context_create(flags, NULL);
    if (speid == NULL) {
        perror("spe_context_create");
        return -2;
    }

    // Load an SPE executable object into the SPE context local store
    if (spe_program_load(speid, &hello_spu)) {
        perror("spe_program_load");
        return -3;
    }

    // Run the SPE context
    rc = spe_context_run(speid, &entry, 0, argp, envp, &stop_info);
    if (rc < 0)
        perror("spe_context_run");

    // Destroy the SPE context
    spe_context_destroy(speid);
    return 0;
}
```

DIRS = spu
PROGRAM_ppu = hello_be1
IMPORTS = spu/hello_spu.a -lspe2 -lpthread
include $(CELL_TOP)/make.footer

PPU Makefile
PPE and SPE Synergistic Programming

#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <libspe2.h>
extern spe_program_handle_t hello_spu;

int main(void)
{
    // Run the SPE context
    rc = spe_context_run(speid, &entry, 0, argp, envp, &stop_info);
    ....
}

#include <stdio.h>

int main(unsigned long long speid, unsigned long long argp, unsigned long long envp)
{
    printf("Hello world!\n");
    return 0;
}
Quick review of the build process and transfer of binaries from host to simulated systems
Build Process

- SPE Source (.c, .cpp) → SPE Compiler → SPE Objects (.o) → Embed SPE Utility
- PPE Source (.c, .cpp) → PPE Compiler → PPE Objects (.o) → PPE Linker

Cell Executable
- PPE Module
- SPE Module
- SPE Module
- SPE Module
- SPE Module

✓ Make scripts are available to automate the build process
Build the code

- **CELL_TOP set to directory containing make header & footer**
  - make.footer contains all the complicated build rules

- **Place SPU code in a subdirectory of directory containing PPC code**
  - e.g. subdirectory name is ‘spu’

- **Makefile for PPC code:**
  - DIRS = spu
  - PROGRAM_ppu = <PPU_executable_name>
  - IMPORTS = <spu_executable-embed.a> -lspe2 -lpthread
  - include $(CELL_TOP)/make.footer

- **Makefile for SPU code:**
  - PROGRAM_spu := <SPU_executable_name>
  - LIBRARY_embed = <spu_executable-embed.a>
  - include $(CELL_TOP)/make.footer
How to Exchange Files between Host and Simulator

- **callthru**
  - A command issued from a simulated windows
  - “backdoor” communication mechanism for the simulated environment to communicate with the host environment
  - Useful for bringing in files to the simulated environment without shutting down and restarting the simulator
  - Example: (binary host → simulator)
    - `callthru source /opt/cell_class/Hands-on/hello_ppu/hello_ppu > hello_ppu`
    - `chmod 755 hello_ppu`
    - `./hello_ppu`
  - Example (result file simulator → host)
    - `callthru sink /home/systemsim-cell/results/result_file < cat result_file`
    - exporting result files out of the simulated environment for later inspection
Running the Binary

- Start the simulator
  - `# cd /opt/ibm/systemsim-cell/run/cell/linux`
  - `#../run_gui`
  - Hit “Go”
Execute Binary

- From the simulated windows, bring executable into the simulator by using the callthru utility, e.g.,
  - `callthru source /opt/cell_class/Hands-on/hello_ppu/hello_ppu > hello_ppu`

- Execute binary
  - `chmod 755 hello_ppu`
  - `./hello_ppu`

Tip!
Copy binary to `/tmp/`<exe> on host to shorten the filename
Building three types of the hello world! program
Directory Structure

/opt/cell_class/Hands-on/ (or /opt/cell_class_/Hands-on-21/hello)
- hello_ppu
- hello_spu
- hello_be1 (hello_be1-sync)
  - spu
Hands-on Exercise

1. Create a directory hello_ppu, write a hello world ppu program and create a Makefile, then compile and execute it as a standalone ppu program

2. Create a directory hello_spu, write a hello world spu program and create a Makefile, then compile and execute it as a standalone spu program

3. Create a directory hello_be1, and write a ppu program that calls an spu program to write hello world. Create all ppu and spu makefiles. Compile and execute those programs to demonstrate the basic structure of a simple PPE-SPE software synergy model (PPE-single SPE model)
Summary

- Understand the basic differences between a ppu, spu, and BE program
- Understand the embedded concept of a cellBE program
- Understand the contents of different Makefile
- Compile and execute different types of cell programs
Special Notices -- Trademarks

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies. All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply. Many of the features described in this document are operating system dependent and may not be available on Linux. For more information, please check: http://www.ibm.com/systems/p/software/whitepapers/linux_overview.html

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.
Special Notices (Cont.) -- Trademarks


A full list of U.S. trademarks owned by IBM may be found at: http://www.ibm.com/legal/copytrade.shtml.

Cell Broadband Engine and Cell Broadband Engine Architecture are trademarks of Sony Computer Entertainment, Inc. in the United States, other countries, or both.
Rambus is a registered trademark of Rambus, Inc.
XDR and FlexIO are trademarks of Rambus, Inc.
UNIX is a registered trademark in the United States, other countries or both.
Linux is a trademark of Linus Torvalds in the United States, other countries or both.
Fedora is a trademark of Redhat, Inc.
Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries or both.
Intel, Intel Xeon, Itanium and Pentium are trademarks or registered trademarks of Intel Corporation in the United States and/or other countries.
AMD Opteron is a trademark of Advanced Micro Devices, Inc.
Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.
TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).
SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPECchpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).
Altivec is a trademark of Freescale Semiconductor, Inc.
PCI-X and PCI Express are registered trademarks of PCI SIG.
InfiniBand™ is a trademark the InfiniBand® Trade Association
Other company, product and service names may be trademarks or service marks of others.

Revised July 23, 2006
Special Notices - Copyrights

(c) Copyright International Business Machines Corporation 2005.
All Rights Reserved. Printed in the United States September 2005.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both.
IBM IBM Logo Power Architecture

Other company, product and service names may be trademarks or service marks of others.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in applications such as implantation, life support, or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page is http://www.ibm.com
The IBM Microelectronics Division home page is http://www.chips.ibm.com