

Linux on the Cell Broadband Engine Architecture

Jeremy Kerr

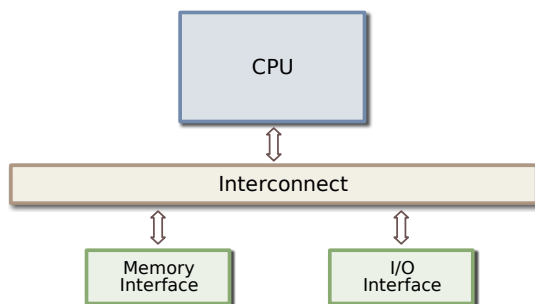
Linux on Cell Kernel Team
IBM Linux Technology Center



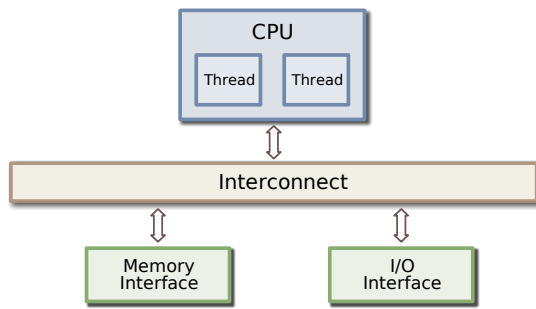
The Architecture



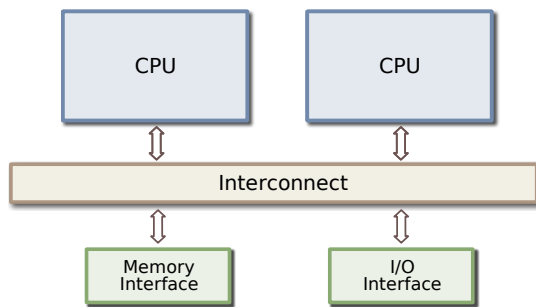
Single CPU



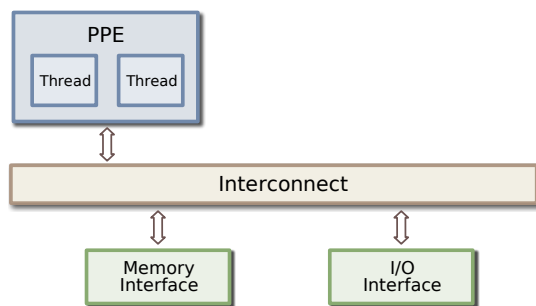
Dual Threading



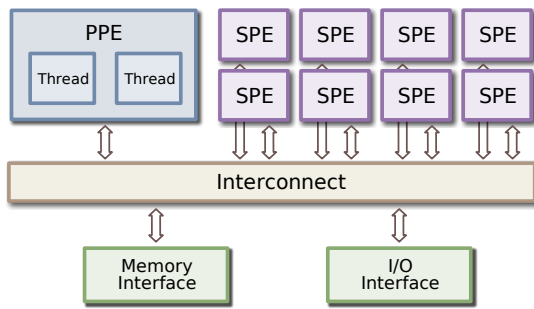
Dual Cores



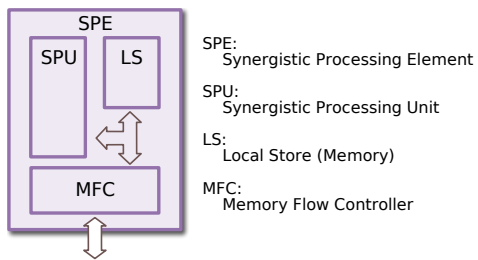
Cell BE CPU



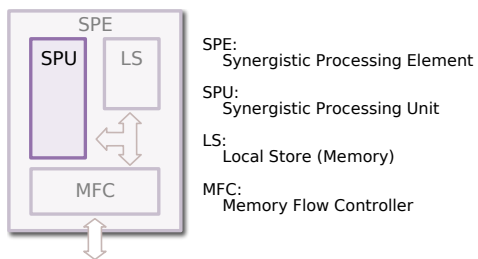
Cell BE CPU



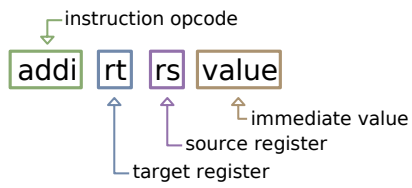
The SPEs



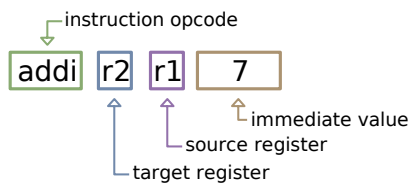
The SPEs



One Instruction



One Instruction



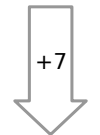
Scalar Instruction

single instruction: addi r2,r1,7

r1 (32 bits)

value (7)

r2 (32 bits)

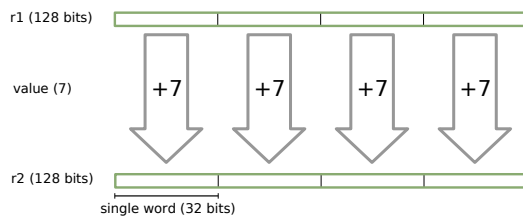


single word

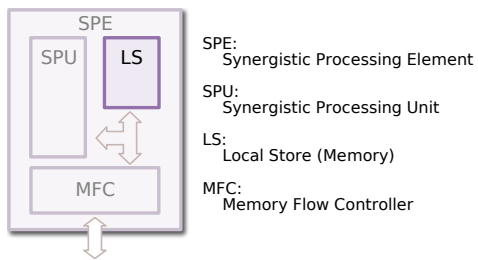


Vector Instruction

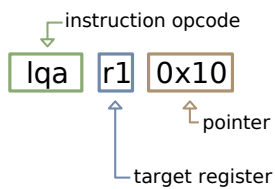
single instruction: `addi r2,r1,7`



The SPEs



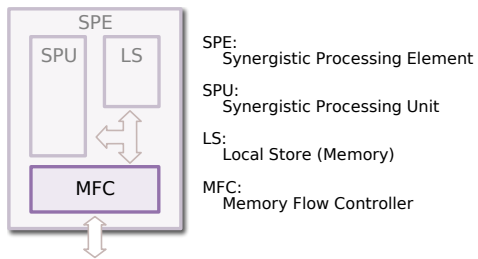
Load Instruction



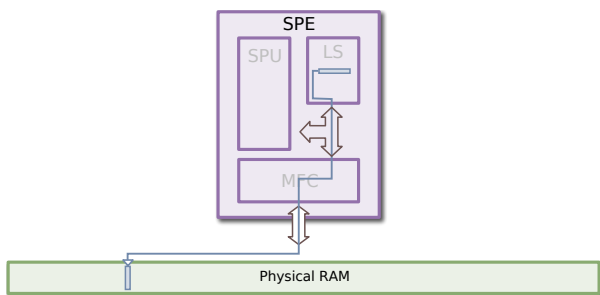
“Take the quadword at **local store** address 0x10, and load it into register 1”



The SPEs



SPE DMA

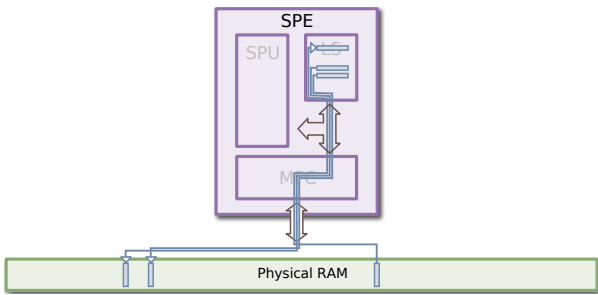


SPE DMA

- Properties of each DMA:
 - Opcode (eg. GET or PUT)
 - Local Store address
 - Main memory address
 - Transfer size
 - Tag ID



SPE DMA



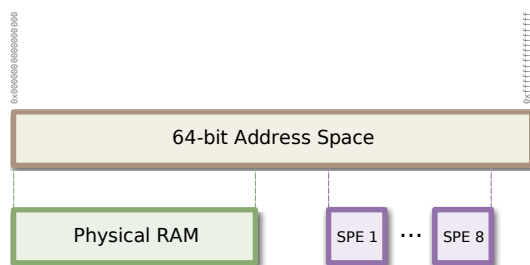
SPE Mailboxes

- Allow simple transfer of 32-bit integers

Mailbox	Direction
Inbound	PPE to SPE
Outbound	SPE to PPE
Outbound Interrupting	SPE to PPE



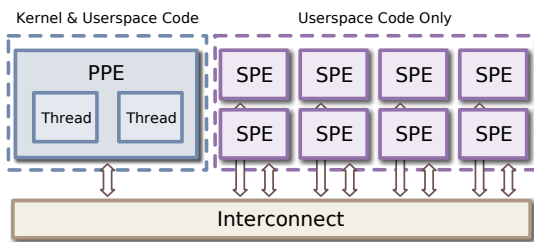
SPE MMIO Control



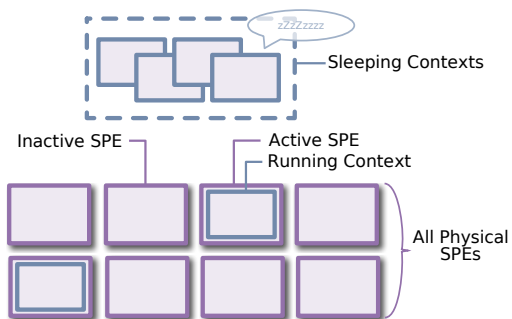
Linux on Cell



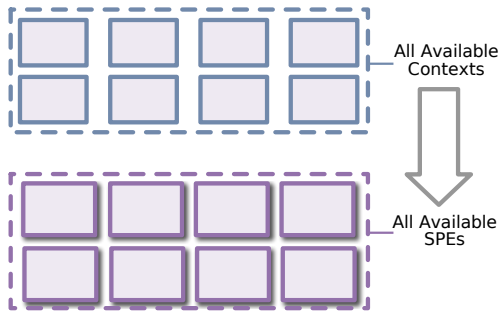
Code Execution



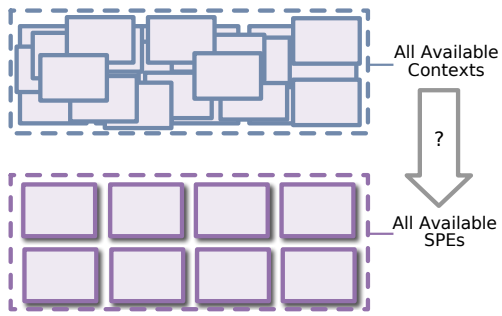
SPE Contexts



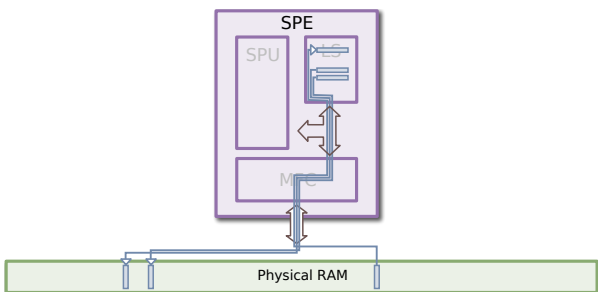
SPE Context Scheduling



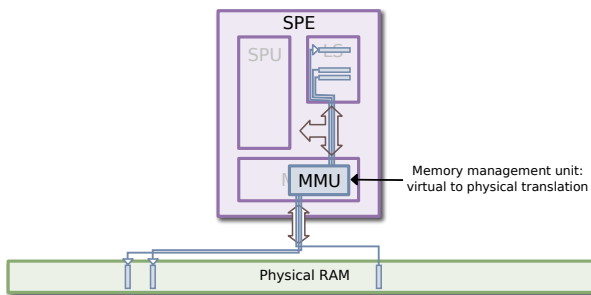
SPE Context Scheduling



Memory Management



Memory Management



Programming Interface



SPE "Hello World"

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



Running SPE Contexts

```
extern spe_program_handle_t spe_program;

int main(void)
{
    spe_context_ptr_t ctx;
    int entry = SPE_DEFAULT_ENTRY;

    ctx = spe_context_create(0, NULL);

    spe_program_load(ctx, &spe_program);

    spe_context_run(ctx, &entry, 0, 0, 0, NULL);
    return 0;
}
```



SPE Mailboxes

•PPE-side:

```
spe_write_in_mbox(ctx, data);
```

•SPE-side:

```
data = spu_readch(SPU_RdInMbox);
```



SPE DMA

•Start:

```
spu_mfcdma32(&local_buf, remote_addr,
             size, tag, MFC_GET_CMD);
```

•Wait for completion:

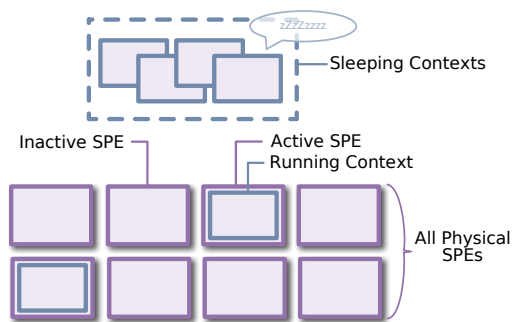
```
mfc_write_tag_mask(1 << tag);
spu_mfcstat();
```



OS Interface



SPE Contexts



Creating Contexts

New system call:

```
int spu_create(const char *path,  
              unsigned int flags,  
              mode_t mode)
```

- Creates a context at the specified path
- Returns a file descriptor for the context



Running Contexts

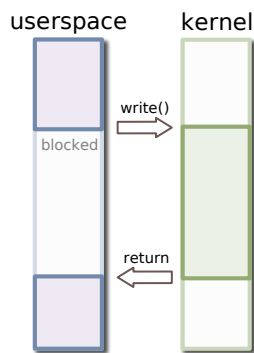
New system call:

```
int spu_run(int fd,  
            u32 *npc,  
            u32 *status)
```

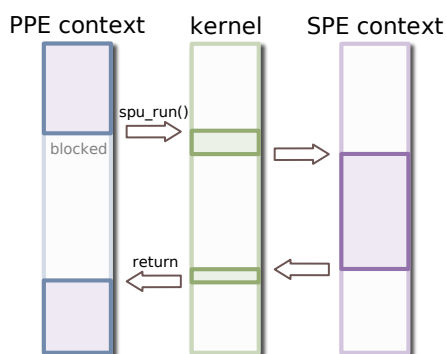
- Finds an available SPE, and runs the context
- Returns the status of the SPE at program stop



System Calls



spu_run System Call



New Filesystem: spufs

```
[jk@cell ~]$ mount
/dev/sda1 on / type ext3 (rw)
sysfs on /sys type sysfs (rw)
proc on /proc type proc (rw)
spufs on /spu type spufs (rw)
```



New Filesystem: spufs

```
[jk@cell ~]$ ls /spu/
spethread-16056-160
spethread-16056-268
spethread-16056-304
spethread-16056-440
spethread-16056-584
spethread-16056-864
```



New Filesystem: spufs

```
[jk@cell ~]$ ls /spu/spethread-16056-160/
cntl          mbox          regs
decr          mbox_info    signal1
decr_status   mbox_stat    signal1_type
dma_info      mem          signal2
event_mask    mfc          signal2_type
event_status  mss          srr0
fpcr          npc          wbox
ibox          object-id    wbox_info
ibox_info     phys-id      wbox_stat
ibox_stat     proxydma_info
lslr         psmap
```



spufs Files: mem

```
[jk@cell ~]$ ls -l /spu/spethread-16056-160/mem
-rw----- 1 jk jk 262144 2008-09-22 10:08 mem
```

- Access to SPE local store
- Appears as a regular file
- eg. read(), write() system calls



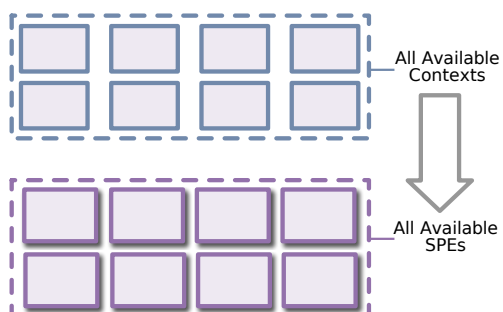
spufs Files: wbox, rbox, ibox

```
[jk@cell ~]$ ls -l /spu/spethread-16056-160/*box
-r----- 1 jk jk 0 2008-09-22 10:08 ibox
-r----- 1 jk jk 0 2008-09-22 10:08 mbox
--w----- 1 jk jk 0 2008-09-22 10:08 wbox
```

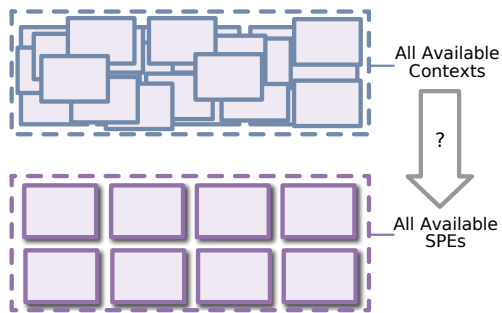
- Access to SPE mailboxes
- Read/write access depends on type



SPE Context Scheduling



SPE Context Scheduling



Resources

- IBM DeveloperWorks
 - <http://ibm.com/developerworks/power/cell>
- Barcelona Supercomputing Center
 - <http://bsc.es/projects/deepcomputing/linuxoncell>



Legal

- This work represents the view of the authors and does not necessarily represent the view of IBM.
- Linux is a registered trademark of Linus Torvalds.
- Cell Broadband Engine is a trademark of Sony Computer Entertainment Inc.
- Other company, product, and service names may be trademarks or service marks of others.

