PS3 Programming

Week 5. Software Cache, Overlay, and SPU Isolation. (Chap 14)

Outline

- Software cache
- Overlay
- SPU Isolation
- Homework

SOFTWARE CACHE

Software cache

- LS (local store) is NOT cache.
- Advantages:
  - Data transfer is much simpler
  - When more data needed than LS can hold.
- Three steps
  - Insert `#define` to define cache parameters
  - Include `cache-apt.h` after the `#define`
  - Using software cache read/write functions
    - Safe and unsafe functions

Configure the cache

- CACHE_NAME: any string
- CACHE_TYPE: 0 (read-only) 1 (read-write)
- CACHED_TYPE: any data type
- CACHE_LOG2NWAY: 0/2 (1/4 way associated)
- CACHE_LOG2NSET: $\log_2$(no of sets)
- CACHELINE_LOG2SIZE: $\log_2$(cache line size)
- CACHE_SET_TAGID(set): tag
- CACHE_READ_X4: enable
- CACHE_STATS: enable monitoring

Safe software cache functions

- Allow to read/write the cache, but don't allow pointer access to the LS

cache_rd(name, ea)	CACHED_TYPE	Read from ea
cache_wr(name, ea, val)	void	Write val to ea
cache_flush(name)	void	Write back
cache_rd_x4(name, ea)	Vector unsigned int	Read a vector from cache

Unsafe software cache functions

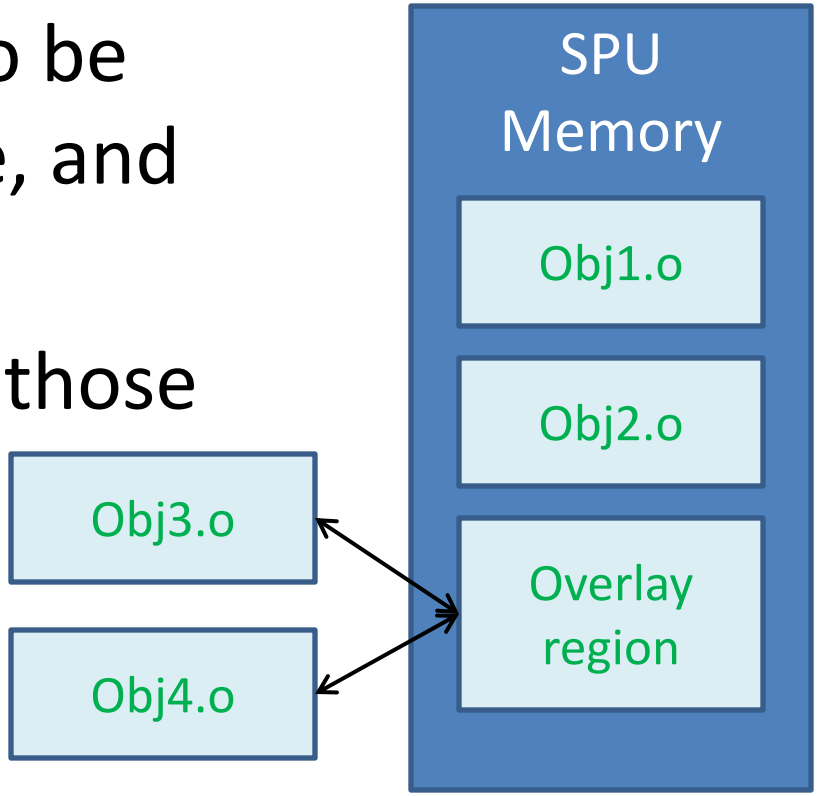
- Return pointers to data in the LS

cache_rw(name, ea)	CACHED_TYPE*	Return the pointer (block)
cache_touch(name, ea)	CACHED_TYPE*	Return the pointer (no block)
cache_wait(name,lsa)	void	Write back
cache_lock(name,lsa)	void	Lock cache line
cache_unlock(name,lsa)	void	Unlock cache line

OVERLAY

Overlay: software paging

- To load a program larger than 256K, you need to use overlay.
- Declare the function to be overlaid in another file, and compiled separately.
- In SPU's main, declare those functions "extern".
- Define a linker script.



SPU program

```
#include <stdio.h>
#include <spu_intrinsics.h>
/* Declare functions in overlay1.o and overlay2.o */
extern vector unsigned int
   reverse_vector(vector unsigned int);
extern vector unsigned int
   reverse_again(vector unsigned int);

int main(unsigned long long speid, unsigned long long
argp, unsigned long long envp) {
   vector unsigned int test_vec =
      (vector unsigned int){0, 1, 2, 3};
   /* Call the external functions */
   test_vec = reverse_vector(test_vec);
   test_vec = reverse_again(test_vec);
   return 0;
}
```

overlay1.c and overlay2.c

```
#include <stdio.h>
#include <spu_intrinsics.h>

vector unsigned int
reverse_vector(vector unsigned int
test){
   int i;
   vector unsigned char indexVec =
{12,13,14,15,8,9,10,11,4,5,6,7,0,1
,2,3};

   /* Rearrange the vector */
   test=spu_shuffle(test,test,
                    indexVec);

   /* Display the results */
   printf("reverse_vector: ");
   for(i=0; i<4; i++)
   printf("%u ",
          spu_extract(test,i));
   return test;
}
```

```
#include <stdio.h>
#include <spu_intrinsics.h>

vector unsigned int
reverse_again(vector unsigned int
test){
   int i;
   vector unsigned char indexVec =
{12, 13, 14, 15,8, 9, 10, 11, 4, 5,
6, 7, 0, 1, 2, 3};

   /* Rearrange the vector */
   test = spu_shuffle(test, test,
indexVec);

   /* Display the results */
   printf("reverse_again: ");
   for(i=0; i<4; i++)
   printf("%u ",
          spu_extract(test, i));
   return test;
}
```

GNU linker script

- The spu linker: spu-ld runs with a script
  - The default script should be in the directory /usr/spu/lib/ldscripts/elf32_spu.x
- Add the following statements to the script

```
OVERLAY :  {
.out_section1 {overlay1.o(.text)}
.out_section2 {overlay2.o(.text)}
}
```

and

```
*( EXCLUDE_FILE(overlay1.o overlay2.o)
          .text .stub .text.* .gnu.linkonce.t.*)
```

Linker flag

- When linking the spu program, run with the flag `LD_FLAGS= -W1, -T, ld.script`
  - Suppose the link script you write is called ld.script

- Put them in the Makefile

```
spu_$(PROJ): spu_$(PROJ).c $(OFILE1).o $(OFILE2).o
	$(SCC) $(LD_FLAGS) -o $@ $^
```

SPU SECURITY AND ISOLATION

SPU security

- Cell protects software by isolating SPUs at the hardware level.
  - Ex: Sony makes one SPU in isolation mode for running Game OS.
- IBM has two versions of SPU security tools
  - Using signed disclosure agreement (CDA) with IBM
  - Part of the Extras in the Cell SDK
    - For testing purpose only

HOMEWORK

Homework

- Try the examples in chap 14
  - Overlay and SPU security
- Try the heapsort in the SDK
  - It uses software cache
  - Read chap 14 for more details
- Continue the compression project