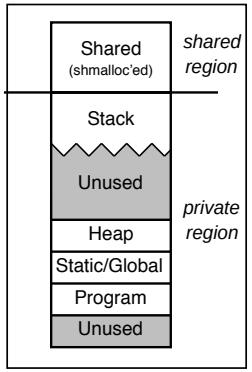
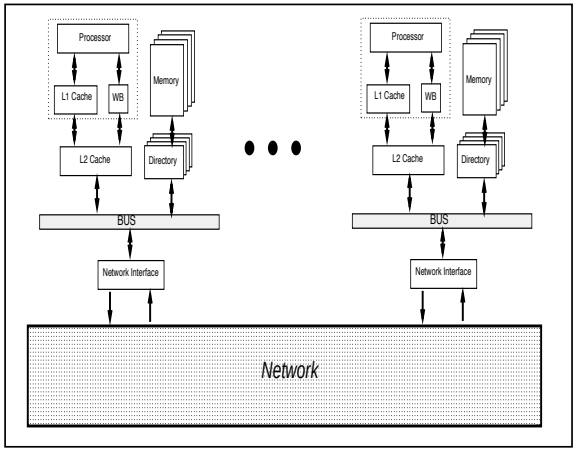
RSIM Introduction

Programming Model



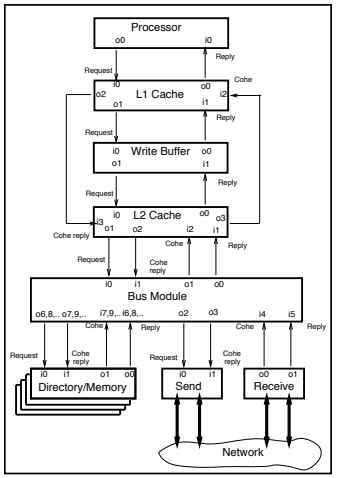
- One "process" per processor
  - Allocated with fork()
  - getpid() returns processor id
- Two-part memory space
  - per-processor part
    - private heap and stack
  - shared heap
    - allocate with shmalloc()

RSIM System Model

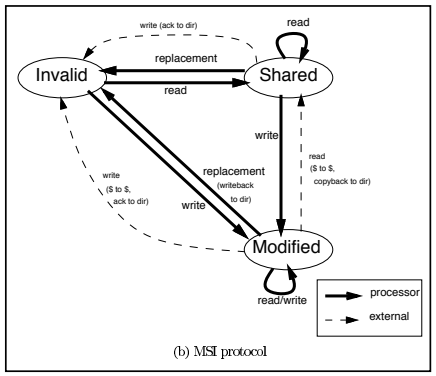


Hierarchical Communication

- Data
  - Requests move down
  - Replies move up
- Coherence
  - Requests move up
  - Replies move down
- E.g., L2 receives and handles data requests from L1, and makes coherence requests to main memory/directories



Your Task



(b) MSI protocol

- Add a 4th state to the coherence protocol between the L2 cache and main memory!
- This is the current protocol (summarized)

What to Change?

- Cache line state machines described by state transition tables
  - Add new state and update transition tables
- L2ProcessTagReq function simulates the interesting part of the L2 cache
- L1ProcessTagReq does the same
- Dir_Cohe function handles coherence messages for the directory

Changing the State Machines

- File: src/MemSys/setup_cohe.c
- Functions:
  - Definitely: setup_secondary_tables()
    - Describes L2 state machine
  - Maybe: setup_tables()
    - Describes L1 state machines

AddEntryToTable

- Adds an entry to the state transition table
- Tables map from (request, curr state) to (next state, action to take on transition)
- The available actions are sending messages higher (closer to processor) or lower (closer to main memory).

Using AddEntryToTable

```
AddEntryToTable(tbl, //the state machine tbl
    req,             // Incoming message
    curr_state,      // Current state
    next_state,      // Next state
    req_mod_next,    // Msg to send on
    req_mod_sz,      // Size of msg
    req_mod_rep_sz,  // Size of reply
    req_next,        // Msg to send back
    req_next_sz,     // Size of msg
    0) // Probably does nothing.
```

Example of AddEntryToTable

```
AddEntryToTable(&Secondary_WB // L2 table
    INVL,            // Invalidate the line
    SH_CL,           // Currently shared
    INVALID,         // Go to INVALID state
    INVL,            // Send INVL to L1
    REQ_SZ,          // Size of INVL
    REQ_SZ,          // Size of reply from
                     // L1 (in future)
    0,               // No msg to dir
    0,               // No msg has no size
    0) // Probably does nothing.
```

Another Example of AddEntryToTable

```
AddEntryToTable(&Secondary_WB // L2 table
    REPL,            // Replace the line
    PR_DY,           // Private, dirty (M)
    INVALID,         // Go to INVALID state
    WRB,             // Send WRB to dir
    REQ_SZ+LINESZ,   // Size of WRB
    REQ_SZ,          // Size of reply from
                     // dir (in future)
    COPYBACK_INVL,   // Message to L1
    REQ_SZ,          // Size of msg to L1
    0) // Probably does nothing.
```

MSHRs

- Miss Status and Handling Registers
- Manage all activity on a cache miss
- Combine outstanding accesses to same line

Getting rsim source

- /cse/courses/cse471/06sp/rsim-cse471.tar
- Use "tar xvf file.tar" to extract contents of file.tar to current working directory
- This creates an rsim-cse471 subdirectory
- We'll call this directory $RSIMHOME

Building rsim

- cd $RSIMHOME/obj/x86
- make
- If no errors, rsim is built in current directory.

Programs in RSIM

- RSIM executes "predecoded sparc binaries"
- Regular sparc binary called program.out
- In same directory is program.dec

Running rsim

- rsim -f program -z conffile -- params to program
- conffile is as described in manual
  - Empty conffile uses default configuration
- program is prefix of program name
  - e.g., "quicksort" runs "quicksort.dec"

Writing rsim programs

- Make a directory for your program, say prg.
- Make subdirectories, prg/src, prg/obj and prg/execs
- Write your program in C, and put the files in the prg/src/ directory
- Copy and modify /cse/courses/cse471/06sp/rsim/apps/QS/makefile
- Tools are in /cse/courses/cse471/06sp/rsim/bin