EXPERIMENTAL operating system

USAGE DOCUMENTATION

Introduction

This documentation describes the usage instructions for SPL Compiler (System Programmer's Language Compiler), APL Compiler (Application Programmer's Language Compiler) , XFS Interface (eXperimental File System Interface) and XSM Simulator (eXperimental String Machine).

top ↑

Setting Up

1. Download the complete package. Go to downloads page.
2. Copy the tar file to your home directory.

```
cp myxos.tar.gz $HOME/
```

3. Extract the contents using the command.

```
tar -xvf myxos.tar.gz
```

   Now you will have a directory `myxos` in your home drectory, with all components required for building your own XOS
4. Change directory to `myxos` directory

```
cd $HOME/myxos
```

5. Make sure all the prerequisites which include gcc, flex/lex and bison/yacc are installed.
   In Ubuntu/Debian systems, use apt to install flex and bison.

```
sudo apt-get install flex bison
```

6. Make to build all the components.

```
make
```

If the setup worked correctly, the following executables will be created

- spl in $HOME/myxos/spl folder
- apl in $HOME/myxos/apl folder
- xfs-interface in $HOME/myxos/xfs-interface folder
- xsm in $HOME/myxos/xsm folder

top ↑

SPL Compiler

Introduction

SPL Compiler (System Programmer's Language Compiler) is used in the implementation of an operating system on XSM (eXperimental String Machine). The compiler compiles the code written in SPL and translates it into machine code which is simulated on the machine. SPL specification is available here.

top ↑

Installation

Prerequisites

- GCC (GNU project C and C++ compiler)
- Flex / Lex (Fast Lexical Analyser Generator)
- Bison / Yacc (GNU Project Parser Generator)

Running
Within your spl directory, use the following commands to run the SPL compiler

- `./spl` *`flag path-to-spl-program`*

Flags

Any one of these flags is required to compile.

- `--os` : Compile OS Code. The compiled output filename will be os_startup.xsm
- `--int=timer` : Compile Timer Interrupt code. The compiled output filename will be timer.xsm
- `--int=[1-7]` : Compile Interrupt routines. The compiled output filename will be int1.xsm, int2.xsm...
- `--exhandler` : Compile Exception Handler. The compiled output filename will be exhandler.xsm

Example Usage

```
cd $HOME/myxos/spl
./spl --os samples/hello-world.spl
```

A file `os_startup.xsm` is created in `$HOME/myxos/spl/samples/`.

top ↑

APL Compiler

Introduction

APL Compiler (Application Programmer's Language Compiler) is used to write programs which can be run on XOS (eXperimental Operating System). The compiler compiles the program written in APL and translates it into machine code which is simulated on the machine. APL specification is available here.

top ↑

Installation

Prerequisites

- GCC (GNU project C and C++ compiler)
- Flex / Lex (Fast Lexical Analyser Generator)
- Bison / Yacc (GNU Project Parser Generator)

Running

Within your APL directory, use the following commands to run the APL compiler

- `./apl path-to-file`

The output file will be in the same location as the input file and filename will be same with the extension changed to `xsm`. For example if input file is prime.apl, output file is prime.xsm

Example Usage

```
cd $HOME/myxos/apl
./apl samples/largest.apl
```

The file largest.apl is a sample APL program which finds largest of 4 numbers. This program is compiled to `$HOME/myxos/apl/samples/largest.xsm`.

top ↑

XFS Interface

Introduction

XFS Interface (eXperimental File System Interface) is an external interface to access the filesystem of the XOS. The filesystem is simulated on a binary file called disk.xfs. The interface can format the disk, load/remove files, list files and copy blocks to a UNIX file. See Filesystem specification.

top ↑

Installation

Prerequisites

- GCC (GNU project C and C++ compiler)

Running
Within your xfs-interface directory, use the following commands to run the interface

- `./xfs-interface`

top ↑

Commands

Type the command help in the interface to display the list of commands.

Format the disk

The command fdisk is used to create the disk ("disk.xfs") or to format the disk if already created. On a newly created disk or formatted disk the Disk Free List and FAT entries are initialized.

Syntax: **`fdisk`**

Load Files

The command load is used to load files from the UNIX filesystem to the XFS disk. The type of the file that is loaded is specified by the first argument. The second argument <pathname> is the path to the UNIX file which is to be loaded to the filesystem.
The command checks the size of the file (executable or data files) and allocates blocks to it. A corresponding FAT entry is created for the file. For the OS startup code, interrupt routines and exception handler, the file is loaded to the corresponding location in the disk (Refer disk organization).

- *Syntax:* **`load --exec <pathname>`**
  Loads an executable file to XFS disk
- *Syntax:* **`load --init <pathname>`**
  Loads INIT code to XFS disk
- *Syntax:* **`load --data <pathname>`**
  Loads a data file to XFS disk
- *Syntax:* **`load --os <pathname>`**
  Loads OS startup code to XFS disk
- *Syntax:* **`load --int=timer <pathname>`**
  Loads Timer Interrupt routine to XFS disk
- *Syntax:* **`load --int=[1-7] <pathname>`**
  Loads the specified Interrupt routine to XFS disk
- *Syntax:* **`load --exhandler <pathname>`**
  Loads exception handler routine to XFS disk

Remove Files

The command rm is used to remove files from the filesystem. The first argument specifies the type of file to be removed. The argument <xfs_filename> specifies the file which is to be removed.
The command searches the FAT entries for the file (executable/data file) and clears the blocks corresponding to the file. In the case of OS startup code, interrupt routines and exception handler the corresponding blocks in the disk are cleared.

- *Syntax:* **`rm --exec <xfs_filename>`**
  Removes an executable file from XFS disk
- *Syntax:* **`rm --init`**
  Removes INIT code from XFS disk
- *Syntax:* **`rm --data <xfs_filename>`**
  Removes a data file from XFS disk
- *Syntax:* **`rm --os`**
  Removes OS startup code from XFS disk
- *Syntax:* **`rm --int=timer`**
  Removes the Timer Interrupt routine from XFS disk

- *Syntax:* **`rm --int=[1-7]`**
  Removes the specified Interrupt routine from XFS disk
- *Syntax:* **`rm --exhandler`**
  Removes the exception handler routine from XFS disk

List Files

The command ls lists all the files which are loaded into the filesystem. The size of the file is also displayed in number of words. The FAT entries are traversed and all the files and their corresponding sizes are displayed.

Syntax: **`ls`**

Display Disk Free List

The command df displays the disk free list. It also displays the total number of blocks and the number of free blocks.

Syntax: **`df`**

Display File contents

The command cat displays the contents of a file in the filesystem. The FAT entries are searched to get the blocks corresponding to the file. The blocks are displayed.

Syntax: **`cat <xfs_filename>`**

Copy disk blocks to a UNIX file

The command copy copies the contents of specified blocks in the filesystem to an external UNIX file. The arguments <start_block> and <end_block> denotes the range of blocks to be copied (including both). <unix_filename> specifes the destination UNIX file to which the contents are copied to.

Syntax: **`copy <start_block> <end_block> <unix_filename>`**

Display help

The command help displays the general syntax and function of all the commands.

Syntax: **`help`**

Exit Interface

The command exit quits the inteface.

Syntax: **`exit`**

Example Usage

To load a file `os_startup.xsm` (which you created using SPL compiler) located in `$HOME/myxos/spl` into the `xfs` filesystem

```
cd $HOME/myxos/xfs-interface
./xfs-interface
```

This will show the `xfs-interface`. First issue a `fdisk` command. This will create a `disk.xfs` file in the directory `$HOME/myxos/xfs-`

`interface`. This file simulates the hard disk for XSM (Machine Simulator).

```
fdisk
```

Now use the `load` command to load the `os_startup.xsm`.

```
load --os $HOME/myxos/spl/samples/os_startup.xsm
```

If you want to see whether load command works properly, copy the block 0 (from 0 to 0) in disk.xfs back to a unix file, say `$HOME/myxos/xfs-interface/test`

```
copy 0 0 $HOME/myxos/xfs-interface/test
```

Now open test to see the contents of os_startup.xsm

top ↑

XSM Simulator

Introduction

The XSM Simulator (eXperimental String Machine) is used to simulate the XSM hardware. See Machine Specification

top ↑

Installation

Prerequisites

- GCC (GNU project C and C++ compiler)
- Flex / Lex (Fast Lexical Analyser Generator)
- Bison / Yacc (GNU Project Parser Generator)

Running
Within your XSM directory, use the following command to run the simulator

- `./xsm` *`flag`*

Flags

- `--debug` : This flag sets the machine into DEBUG mode when it encounters a **BRKP** machine instruction. Further details are given in the section below.
- `--timer=value` : This flag sets the number of instructions after which timer interrupt is triggered to `value`. --timer=0 disables the timer. The range of value is from 0 to 1024

Debugging

The `--debug` flag is used to debug the running machine. When this flag is set and the machine encounters a `breakpoint` instruction, it is set into the DEBUG mode. In this mode a prompt is displayed which allows the user to enter commands to inspect the state of the machine.

The commands in DEBUG mode are

- `step / s` :
  The execution proceeds by a single step
- `continue / c` :
  The execution proceeds till the next `breakpoint` instruction.
- `reg / r` :
  Displays the contents of all the registers in the machine namely `IP, SP, BP, PTBR, PTLR, EFR, R0-R7, S0-S15 and T0-T3`.
  Sample usage: `reg`
- `reg / r <register_name>` :
  Displayes the content of the specified register.
  Sample usage: `r R5, reg PTLR`
- `reg / r <register_name_1> <register_name_1>` :
  Displays the contents of the registers from <register_name_1> to <register_name_1>.
- `mem / m <page_num>` :
  Displays contents of the memory page <page_num>.
  Sample usage: `mem 5, m 20`
- `mem / m <page_num_1> <page_num_2>` :
  Displays the contents of the memory from pages <page_num_1> to <page_num_2>.
  Sample usage: `mem 5 8, m 0 10`
- `pcb / p` :
  Displays the Process Control Block of the process with the state as RUNNING.
- `pcb / p <pid>` :
  Displays the Process Control Block of the process with the given `<pid>`.
- `pagetable / pt` :
  Displays the Page Table at the location pointed by **PTBR** (Page Table Base Register).
- `pagetable / pt <pid>` :
  Displays the <pid>th Page Table.
- `filetable / ft` :
  Displays the System Wide Open File Table.

- `memfreelist / mf` :
  Displays the Memory Free List.
- `diskfreelist / df` :
  Displays the Memory copy of Disk Free List.
- `fat` :
  Displays the memory copy of the File Allocation Table (FAT).
- `location / l <address>` :
  Displays the content at memory address (Address Translation takes place if used in USER mode).
- `watch / w <physical_address>` :
  Sets a watch point to this address. Watch point is used to track changes of a particular memory location. Whenever a word which is watched is altered, debug interface is invoked. Atmost 16 watch points can be set.
- `watchclear / wc` :
  Clears all the watch points.
- `exit / e` :
  Exits the debug prompt and halts the machine.
- `help / h` :
  Displays commands.

top ↑