System Programming
Lab 07: Using GXEmul

Hugo Marcondes
hugom@lisha.ufsc.br
http://www.lisha.ufsc.br/~hugom

Oct 2006

GXEmul

- An experimental instruction-level machine emulator
  - Several emulation modes
    - Just processors
    - Processors + Surrounding hardware
  - Processors not simulated with accurancy
    - Only “fakes” well enough to allow guest OS run without complain
  - Run real code generated with a real compiler
    - Interest for academic research and experiments
    - Support for debugging

Running GXEmul

- Use the emulation mode testmips

  gxemul -E <emulation_mode> <executable>

  Options:
  - -E -> Specify emulation mode
  - -V -> Just load executable file
  - -q -> Quiet mode

  Debugging:
  - breakpoint (add | delete | show) <symbol>
  - continue | step <n>
  - dump [start_address [end_address]]
  - reg
  - put [b|h|w|d|q] addr, data

The testmips Emulation

- Provides basic devices
  - Console Device
  - Linear Frame buffer for graphics output
  - Disk Controller
  - Ethernet Controller
- Physical RAM
  - Emulated at address 0x80030000

Running Application

- GNU GCC/Binutils as cross-compiler
- Installed on /usr/local/mips32/

mips-gcc

- C language compiler
- Use -c to generate an object file
  mips-gcc -c util.c

mips-as

- GNU Assembler
- Don't have the same syntax of SPIM !

MIPS-AS

- Don't provide mnemonics for all registers

```
la $a0, myString          # WRONG !
la $4, myString
```

- Most assembler directives are similar
  - To declare a external symbol:

```
.extern <symbol>
```

- Additional Documentation

http://www.gnu.org/software/binutils/

Linking Object Files

mips-ld – Link several object files

- Options:
  -Ttext <addr> : Allocate segment text in addr
  -e <symbol> : Set the entry point as symbol
  -o <file> : Output file

- Example:
  mips-ld -Ttext 0x80030000 -e main main.o util.o -o main

- Linker Scripts
  - Could be used to drive the allocation of segments on the linking process

Exercise

- Run the exercise of Lab04 on GXEmul

  File util.c provide the following procedures:

```
void printstr(char * str);
void print_int(int i);
void halt();
```

- Don't forget to update the syntax of your assembly code !