CS2610: Computer Organization and Architecture Lab

Working with Data[1]

Madhu Mutyam
PACE Laboratory
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Feb 5, 2019

[1] Igor Zhirkov. Low-Level Programming. Apress, 2017.

Endianness

- Big endian – multibyte numbers are stored in memory starting with the *most* significant byte.
- Little endian – multibyte numbers are stored in memory starting with the *least* significant bytes.

Write an assembly program to check the endianness of your computer.

Strings

- *String* is a sequence of characters.
- A special character (the zero-code) denotes the string ending.

Pointers and Different Addressing Modes

- Pointers are addresses of memory cells.
- The pointer size is 8 bytes.
- Different Addressing Modes:
  - Immediate: `mov rax, 10`
  - Register: `mov rax, rax`
  - Direct: `mov rax, [10]`
  - Register Indirect: `mov rax, [r9]`
  - Base-indexed with scale and displacement:

```
mov rax, [rbx + 4 * rcx + 9]
```

$$Address = base + scale * index + displacement$$

  - *Base* is either immediate or a register.
  - *Scale* can only be immediate, equal to 1, 2, 4, or 8.
  - *Index* is immediate or a register; and
  - *Displacement* is always immediate.

Input/Output Library Functions

Function	Definition
`string_length`	Accepts a pointer to a string and returns its length.
`print_string`	Accepts a pointer to a null-terminated string and prints it to `stdout`.
`print_char`	Accepts a character code directly as its first argument and prints it to `stdout`.
`print_newline`	Prints a character with code `0xA`.
`print_uint`	Outputs an unsigned 8-byte integer in decimal format.
`print_int`	Outputs a signed 8-byte integer in decimal format.
`read_char`	Read one character from `stdin` and return it. If the end of input stream occurs, return 0.
`read_word`	Accepts a buffer address and size as arguments. Reads next word from `stdin` (skipping whitespaces into buffer). Stops and returns 0 if word is too big for the buffer specified; otherwise, returns a buffer address.

Input/Output Library Functions (Contd)

Function	Definition
`parse_uint`	Accepts a null-terminated string and tries to parse an unsigned number from its start. Returns the number parsed in `rax`, its characters count in `rdx`.
`parse_int`	Accepts a null-terminated string and tries to parse a signed number from its start. Returns the number parsed in `rax`, its characters count in `rdx` (including sign if any). No spaces between sign and digits are allowed.
`string_equals`	Accepts two pointers to strings and compares them. Returns 1 if they are equal, otherwise 0.
`string_copy`	Accepts a pointer to a string, a pointer to a buffer, and buffer's length. Copies string to the destination. The destination address is returned if the string fits the buffer; otherwise, zero is returned.
`exit`	Accepts an exit code and terminates current process.

Using gdb

- To launch gdb: `gdb executable_file`
  `(gdb)`

Command	Description
`quit`	To quit gdb
`help cmd`	To show help for the command `cmd`
`run`	Starts program execution
`break x`	Creates a breakpoint near the label `x`
`break *address`	Creates a breakpoint at a specified address
`continue`	Top continue running program
`stepi` or `si`	To step by one instruction
`nexti` or `ni`	Same as above, but will not enter function if the instruction was `call`. It will let the called function terminate and break at the next instruction

Using gdb (Contd)

- `layout asm`
- `layout regs`
- `print /FMT <val>` – to check register contents or memory values. Register names are prefixed with $
- `x /FMT <address>` – to check memory contents
- `FMT` – Encoded Format Description
  - `x` – Hexadecimal
  - `a` – Address
  - `i` – Instruction
  - `c` – Char
  - `s` – Null-terminated string

Thank You