Laboratório de Microprocessadores

Compilação Cruzada

Giovani Gracioli
giovani@lisha.ufsc.br

Março 2010

Roteiro

- Visão geral sobre compilação cruzada
- Exemplos de geração de código
- Conversão de arquivos binários
- Carregamento de executáveis em SE

Compilação Cruzada

- Cross-Compiler é o processo de gerar código através do compilador para uma plataforma diferente daquela que o compilador está executando
- Além do compilador, o ambiente de compilação cruzada é formado por diversas ferramentas (GNU binutils), que servem para manipular código objeto em diferentes formatos

Exemplos de Ferramentas

- **as**: montador
- **ld**: ligador
- **ar**: cria, modifica e extrai informações de "archives"
- **objcopy**: cópia de arquivos objetos, geralmente fazendo modificações
- **size**: lista o tamanho das seções
- **strip**: remove símbolos do arquivo objeto
- **objdump**: informações sobre os arquivos objetos
- **gdb**, etc

Porque compilação cruzada?

- Uso fundamental é separar o ambiente de programação do ambiente ou plataforma alvo:
  - **Sistemas embarcados:** recursos limitados
  - **Múltiplas máquinas:** diferentes versões de um software para diferentes plataforma
  - **Processo de inicialização para uma nova plataforma:** gerar software necessário para iniciar a nova plataforma (SO, compilador, etc)
- E as máquinas virtuais?

Geração de um ambiente

- Passos para configurar o ambiente de compilação cruzada:
  1) Usar o compilador nativo para gerar um compilador da arquitetura da plataforma alvo
  2) Gerar as ferramentas necessárias para a plataforma alvo
  3) Usar o novo compilador para gerar código para a arquitetura alvo
  4) Verificar as ferramentas necessárias para carregar o executável
  5) Carregar o executável para a plataforma

Exemplo: gerar um novo gcc

- Binutils e gcc compilados com a mesma opção –target=some-target no script de configuração
- Conceito de build platform, host platform e target platform
  - Build platform: onde o código é compilado
  - Host platform: onde o código é executado
  - Target platform: aplicável somente em compiladores, representa o tipo de código objeto que será gerado

Exemplo: código para IA32 x AVR8

- Dado o código abaixo:

```
int main(void) {
    printf(“hello world\n”);
}
```

- Compile para a arquitetura IA32
  - gcc -c test1.o
  - file test1.o
- Analise o assembly gerado
  - objdump -dS test1.o | less
- Gere o executável
  - gcc -o test1 test1.o
  - file test1
  - objdump -dS test1 | less
- O que mudou? porque?

Exemplo: código para IA32 x AVR8

- Utilize o mesmo exemplo, mas agora use o compilador para a arquitetura AVR8:
  - avr-gcc -c test1.c
  - file test1.o
  - avr-objdump -dS test1.o | less
  - avr-gcc -o test1 test1.o
  - avr-objdump -dS test1 | less
- O que mudou? porque?

Exemplo: código para IA32 x AVR8

- Repita o mesmo procedimento anterior, mas agora passe as seguintes flags de compilação:
  - -mmcu=at90s8515
  - -mmcu=atmega128
  - -mmcu=atmega16
  - -mmcu=at90can128
- Gere os assemblys de cada um com avr-objdump
- Porque o símbolo __vectors muda?
- Dica: baixe os datasheets dos microcontroladores

Converter o executável

- ELF é o padrão de arquivos objetos usado atualmente na maioria dos sistemas. Porém, sistemas embarcados os formatos de arquivos objetos tendem a serem mais simples, como SREC e IHEX
- É necessário verificar qual o formato aceito pela plataforma alvo
  - Para STK500 e mica2, por exemplo, o formato é o IHEX
  - avr-objcopy -O ihex elf-entrada saida.hex
  - formato texto: cat saida.hex

Carregar o executável

- Para carregar o executável gerado na plataforma alvo, é necessário uma ferramenta específica, geralmente chamada de loader, programador flash ou monitor
- Para o STK500, pode-se usar o UISP (AVR's In-System Programmer)
- Ou ainda o avr-dude
- Com a mesma ferramenta é possível executar diversas tarefas:
  - Apagar a memória
  - Configurar os fuse bits, etc

Carregando o primeiro executável

- Para carregar o arquivo hex no stk500
  - avrdude -P /dev/ttyUSB0 -c stk500v2 -p **at90s8515** -U flash:w:file.hex -C /usr/local/avr/tools/etc/avrdude.conf -F
- Para carregar no mica2
  - uisp -dprog=mib510 -dpart=atmega128 -dserial=/dev/ttyUSB0 --erase --upload if=file.hex
- Porque printf não funciona?
  - Dica: procure os passos para inicializar a UART no datasheet

Exercício: piscar leds e UART

- Baixe o arquivo leds.c da página da disciplina
  - Porte o arquivo para o ATMega128
  - Dica: verifique os endereços das portas A e B
- Implemente uma função que imprima um caracter pela UART