Linguagens de Programação
Prof. Andrei Rimsa Álvares

Lista de Exercícios II

Sistemas de Linguagens

Exercício 01) O programa **gcc** presente em sistemas Unix para invocar o compilador de C desenvolvido pela *GNU* é na verdade um script que invoca vários outros programas. É possível descobrir a cadeia de programas invocados por **gcc** adicionando o parâmetro **–v** à linha de comando. Cada um dos itens a seguir correspondem a um dos passos adotados por **gcc** para produzir um arquivo binário a partir de um programa fonte. Descreva o que cada linha faz:

a) `gcc –E hello.c -o hello_p.c`
b) `gcc –S hello_p.c –o hello_p.s`
c) `as hello_p.s –o hello.o`
d) `ld hello.o –o a.out`

Exercício 02) Para cada uma das variações da sequência clássica indique duas vantagens e duas desvantagem. Liste uma linguagem que usa essa variação.

a) Interpretação
b) Máquinas virtuais
c) Ligação tardia
d) Instrumentação de código (profiling)
e) Compilação dinâmica (JIT: Just In-Time)

Exercício 03) Um compilador *just-in-time* (*JIT*) compila os programas enquanto os mesmos ainda estão sendo interpretados. O programa a seguir, escrito em C, representa um compilador JIT muito rudimentar.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>

int main(void) {
	char* program;
	int (*fnptr)(void);
	int a;
	program = mmap(NULL, 1000, PROT_EXEC | PROT_READ | PROT_WRITE,
		MAP_PRIVATE | MAP_ANONYMOUS, 0, 0);
	program[0] = 0xB8;
	program[1] = 0x34;
	program[2] = 0x12;
	program[3] = 0;
	program[4] = 0;
	program[5] = 0xC3;
	fnptr = (int (*)(void)) program;
	a = fnptr();
	printf("Result = %X\n", a);
}
```

Responda as questões a seguir baseado no programa acima:

a) Compile esse programa e diga o que será impresso por ele.
b) Qual é o "programa" produzido por esse compilador JIT? Onde esse "programa" é armazenado?
c) Esse programa funcionaria em uma arquitetura diferente de x86? Por quê?

Exercício 04) Compiladores modernos são muito bons em otimizar código. Contudo, algumas linguagens de programação permitem que o desenvolvedor indique como esses programas podem ser otimizados.

a) Considere a palavra-chave **`register`** da linguagem C. Para que ela serve e de que forma ela permite que o desenvolvedor "guie" o otimizador de código? Ilustre sua resposta com um trecho de código.
b) Responda as mesmas perguntas da questão anterior, porém dessa vez com a palavra-chave **`inline`** em C++.

Exercício 05) Existem diversos depuradores de código. Dois famosos, usados para C/C++, são **`gdb`** e **`valgrind`**. Esses depuradores têm propósitos muito diferentes conforme será visto a seguir.

a) A ferramenta **`gdb`** possibilita uma execução "interativa" de um programa. Para que isso funcione bem, é preciso compilar o programa com o parâmetro `-g`. Quando isso acontece, nota-se que o código binário produzido cresce. Por quê acontece tal crescimento? A titulo de exemplo:

```
$ g++ -o prog-nodbg prog.cpp
$ g++ -g -o prog-dbg prog.cpp
$ ls -l prog-*
-rwxr-xr-x  1 rimsa  staff  69752 Oct  4 22:05 prog-dbg
-rwxr-xr-x  1 rimsa  staff  59952 Oct  4 22:05 prog-nodbg
$
```

b) A ferramenta **`valgrind`** permite depurar problemas de memória como vazamentos de espaço alocado, popularmente conhecidos como *memory leaks*. Qual a saída produzida para o **`valgrind`** para o programa abaixo (execute como **`valgrind ./a.out`**)? E se fosse removido o comentário sobre o comando free?

```c
#include <stdio.h>
#include <stdlib.h>

void problem() {
	int* i = (int*) malloc(sizeof(int));
	*i = 3;
	printf("%d\n", *i);
	// free (i);
}
int main() {
	problem();
}
```

Linguagens de Programação
Prof. Andrei Rimsa Álvares