Lista de Linguagens de Programação – 4

Nome: _____ Matrícula: _____

1. O programa `gcc`, usado em Unix para invocar o compilador de C desenvolvido pela gnu, é na verdade um *script* que invoca vários outros programas. É possível saber quais os programas invocados por `gcc` adicionando-se o parâmetro `-v` à sua linha de invocação. Cada uma das questões abaixo contém um dos passos adotados por `gcc` para produzir um arquivo binário a partir de um programa fonte `hello.c`. Descreva o que cada uma destas linhas faz.

   (a) `gcc -E hello.c > hello.p.c`

   (b) `gcc -S hello.p.c -o hello.p.s`

   (c) `as hello.p.s -o hello.o`

   (d) `/usr/bin/ld hello.o -o a.out`

2. Para resolver este exercício, é necessário estudar as transparências da aula. Há um *link* para as transparências a partir da página do curso. Em Java, qual é o tempo de ligação dos seguintes construtos?

   (a) A localização em memória de uma variável local em um método.

   (b) A localização em memória de um campo não estático de uma classe.

   (c) O significado da palavra *while*.

   (d) O tamanho em bits de um valor to tipo inteiro.

   (e) Os *bytecodes* que formam uma classe.

   (f) A definição do método `m`, quando uma chamada `a.m()` é executada.

   (g) O tipo de uma variável local em um método.

   (h) Os valores atribuídos à uma variável.

   (i) O tamanho de uma referência.

3. Um compilador JIT (do inglês *just-in-time*) compila os programas enquanto os mesmos ainda estão sendo interpretados. O programa escrito em C abaixo representa [1] um compilador JIT muito rudimentar. Responda as questões seguintes sobre este programa.

```
#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);
}
```

(a) Compile este programa e diga o que será impresso por ele.

(b) Qual é o "programa" produzido por este compilador JIT? Onde este "programa" é armazenado?

(c) Este programa funcionaria em alguma arquitetura diferente de x86? Por que?

[1] Contribuição do aluno Dilson Guimarães.

(d) Linguagens como JavaScript em geral se beneficiam mais de compiladores JIT que linguagens como Java, embora ambas usem esta técnica. Por que?

4. Compiladores modernos são muito bons em otimizar programas. Algumas linguagens de programação, contudo, permitem que o desenvolvedor explique para o compilador como estes programas devem ser otimizados.

   (a) Considere a palavra chave `register`, na linguagem C. Para que serve esta palavra chave, e de que forma ela permite que o desenvolvedor "guie" o otimizador de código? Ilustre a sua resposta com um trecho de código.

   (b) Responda às mesmas perguntas da questão anterior, porém desta vez para a palavra chave `inline` em C++.

5. Existem diversos tipos de depuradores de código. Dois dos mais famosos, usados em C e C++, são `gdb` e `valgrind`. Estes depuradores têm propositos muito diferentes conforme veremos a seguir.

   (a) A ferramenta `gdb` possibilita uma execução "interativa" de um programa. Para que `gdb` funcione bem, é preciso compilar o programa alvo com o parâmetro `-g`. Quando isto acontece, notamos que o código binário produzido cresce um pouco. Por que se dá tal crescimento? A título de exemplo:

```
$> gcc opt.c -o optNoDebug
$> gcc -g opt.c -o optDebug
$> ls -la opt*Debug
-rwxr-xr-x  1 fpereira  staff  13000 Mar  9 21:46 optDebug
-rwxr-xr-x  1 fpereira  staff  12644 Mar  9 21:46 optNoDebug
```

   (b) A ferramenta `valgrind` permite depurar problemas de memória, como os vazamentos de espaço alocado, popularment conhecidos como *memory leaks*.

      i. Qual a saída produzida por `valgrind` para o programa abaixo? [2]

```
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();
}
```

      ii. Qual a saída produzida por `valgrind` caso removêssemos o comentário sobre o comando `free`?

[2] No linux, use `valgrind -v ./a.out`.