Universidade Estadual de Santa Cruz – UESC

Relatório p-code machine utilizando fibonacci e factorial recursivos como exemplos

Relatório feito por Darley Souza Sampaio

Disciplina Compiladores

Curso Ciência da Computação

Semestre 2022.2

Professor César Alberto Bravo Pariente

Ilhéus – BA
2022

ÍNDICE

Configurações da Máquina

Para fins de conhecimento da máquina utilizada, segue as seguintes configurações onde o código foi transcrito e compilado:

Processador: AMD Ryzen 3200G

Placa de Vídeo: Geforce GTX 1050TI 4GB OC

Placa Mãe: Asrock A320M-HD

Memória: 16GB (2x8) DDR4 2666mHz

Armazenamento: SSD M.2 120GB + HD 1TB

Sistema Operacional: Linux(Mint)

Comandos de Operação

Comandos para realizar as operações.

Comando	Ação
LIT 0 a	armazena valor em a no topo da pilha
OPR 0 a	executa uma operação selecionada por a
LOD l a	carrega a variável a de l
STO l a	salva a variável a em l
CAL l a	chama o procedimento de nível l em a
INT 0 a	adiciona ao registro um tamanho a
JMP 0 a	pula para a
JPC 0 a	pulo condicional para a

Códigos da Operação OPR

Códigos utilizados dentro da operação OPR para realizar as ações.

Código	Símbolo	Ação
0	Return	Return from a subroutine
1	Negate	Negate the value at the top of the stack
2	ADD	x = pop(); y = pop(); push(y + x)
3	Subtract	x = pop(); y = pop(); push(y - x)
4	Multiply	x = pop(); y = pop(); push(y * x)
5	Divide	x = pop(); y = pop(); push(y / x)
6	ODD?	Checks the top value of the stack whether it is odd or not
7	==	x = pop(); y = pop(); push(y == x)
8	< >	x = pop(); y = pop(); push(y < > x)
9	<	x = pop(); y = pop(); push(y < x)
10	>=	x = pop(); y = pop(); push(y >= x)
11	>	x = pop(); y = pop(); push(y > x)
12	<=	x = pop(); y = pop(); push(y <= x)

Compilando o Programa

Para realizar a compilação do programa basta inserir em qualquer compilador. Caso esteja utilizando o linux, utilize os seguintes comandos em seu terminal.

O link para download de todo o projeto em sí é: https://github.com/danssampaio/compilers/tree/main/p-code_machine/Proj1d

Realiza a compilação e cria um arquivo executável.
$ g++ -o p-code p-code.cpp

Inicia o arquivo executável através do terminal.
$ ./p-code

Questões Utilizadas

Para realizar testes nesse programa, foram utilizadas duas sequências matemáticas solicitadas em sala de aula, o fibonacci de 5 e o factorial de 4 ambos recursivos.

Fibonacci 5

```
INT 0 3
LIT 0 5
STO 0 6
CAL 0 6
LOD 0 6
OPR 0 0
INT 0 5
LOD 0 3
LIT 0 1
OPR 0 12
JPC 0 13
STO 0 4
OPR 0 0
LOD 0 3
LIT 0 1
OPR 0 3
STO 0 8
CAL 0 6
LOD 0 8
LOD 0 13
OPR 0 2
STO 0 3
OPR 0 0
```

```
// main()

code[0].f = INT; code[0].l = 0; code[0].a = 3;        // Ram[3] Pilha[3]

code[1].f = LIT; code[1].l = 0; code[1].a = 5;        //Fibonacci à se calcular

code[2].f = STO; code[2].l = 0; code[2].a = 6;        //Salva como parâmetro

code[3].f = CAL; code[3].l = 0; code[3].a = 6;        //Chamando a função fib()

code[4].f = LOD; code[4].l = 0; code[4].a = 6;        //Return fib

code[5].f = OPR; code[5].l = 0; code[5].a = 0;        // Return 0


//fib()

code[6].f = INT; code[6].l = 0; code[6].a = 5;        // Ram[5] Pilha[5]

code[7].f = LOD; code[7].l = 0; code[7].a = 3;        // Load parâmentro

code[8].f = LIT; code[8].l = 0; code[8].a = 1;        // Sobe 1 para comparação

code[9].f = OPR; code[9].l = 0; code[9].a = 12;       // Parâmetro <= 1 ??

code[10].f = JPC; code[10].l = 0; code[10].a = 13;
```

```
//if(n <= 1)
code[11].f = STO; code[11].l = 0; code[11].a = 4;      //Return = resultado
code[12].f = OPR; code[12].l = 0; code[12].a = 0;      // OPR 0 0

//else
code[13].f = LOD; code[13].l = 0; code[13].a = 3;       // Load parâmetro
code[14].f = LIT; code[14].l = 0; code[14].a = 1;       // Sobe 1 para subtração
code[15].f = OPR; code[15].l = 0; code[15].a = 3;       // Parâmetro - 1
code[16].f = STO; code[16].l = 0; code[16].a = 8;       // Salva como parâmetro
code[17].f = CAL; code[17].l = 0; code[17].a = 6;       // Chama a função fib()

//sum
code[18].f = LOD; code[18].l = 0; code[18].a = 8;       // Load parâmetro
code[19].f = LOD; code[19].l = 0; code[19].a = 13;       // Load parâmetro 2
code[20].f = OPR; code[20].l = 0; code[20].a = 2;       // sum top retorno
code[21].f = STO; code[21].l = 0; code[21].a = 3;       // Salva no return
code[22].f = OPR; code[22].l = 0; code[22].a = 0;       // OPR 0 0
```

OUTPUT: 5

Factorial 4

```
INT 0 5
LIT 0 4
STO 0 8
CAL 0 6
LOD 0 9
OPR 0 0
INT 0 5
LOD 0 3
LIT 0 1
OPR 0 12
JPC 0 13
STO 0 4
OPR 0 0
LOD 0 3
LIT 0 1
OPR 0 3
STO 0 8
CAL 0 6
LOD 0 3
LOD 0 9
OPR 0 4
STO 0 4
OPR 0 0
```

```
// main()
code[0].f = INT; code[0].l = 0; code[0].a = 5;        // Ram[5] Pilha[5]
code[1].f = LIT; code[1].l = 0; code[1].a = 4;        // Fatorial à se calcular
code[2].f = STO; code[2].l = 0; code[2].a = 8;        // Salva como parâmetro
code[3].f = CAL; code[3].l = 0; code[3].a = 6;        // Chamando a função fat()
code[4].f = LOD; code[4].l = 0; code[4].a = 9;        // Return fat()
code[5].f = OPR; code[5].l = 0; code[5].a = 0;        // Return 0

//fat()
code[6].f = INT; code[6].l = 0; code[6].a = 5;        // Ram[5] Pilha[5]
code[7].f = LOD; code[7].l = 0; code[7].a = 3;        // Load parâmentro
code[8].f = LIT; code[8].l = 0; code[8].a = 1;        // Sobe 1 para comparação
code[9].f = OPR; code[9].l = 0; code[9].a = 12;       // Parâmetro <= 1 ??
code[10].f = JPC; code[10].l = 0; code[10].a = 13;
```

```
//if(n <= 1)
code[11].f = STO; code[11].l = 0; code[11].a = 4;      // Return = resultado
code[12].f = OPR; code[12].l = 0; code[12].a = 0;      // OPR 0 0

//else
code[13].f = LOD; code[13].l = 0; code[13].a = 3;      // Load parâmetro
code[14].f = LIT; code[14].l = 0; code[14].a = 1;      // Sobe 1 para subtração
code[15].f = OPR; code[15].l = 0; code[15].a = 3;      // Parâmetro - 1
code[16].f = STO; code[16].l = 0; code[16].a = 8;      // Salva como parâmetro
code[17].f = CAL; code[17].l = 0; code[17].a = 6;      // Chama a função fat()

//mult
code[18].f = LOD; code[18].l = 0; code[18].a = 3;      // Load parâmetro
code[19].f = LOD; code[19].l = 0; code[19].a = 9;      // Load Return
code[20].f = OPR; code[20].l = 0; code[20].a = 4;      // Multiplica
code[21].f = STO; code[21].l = 0; code[21].a = 4;      // Salva no return
code[22].f = OPR; code[22].l = 0; code[22].a = 0;      // OPR 0 0
```

OUTPUT: 24

Considerações Finais

Durante a realização dos testes dos exemplos utilizados neste programa, obtive dificuldade em encontrar o método de retornar os valores recursivos. Diante desse fato, faz-se necessário um estudo mais aprofundado do funcionamento de um programa recursivo em um p-code da minha parte.

Referências

https://en.wikipedia.org/wiki/P-code_machine

http://th.cpp.sh/9nsyz

https://trendspdf.prograd.uesc.br/MaterialApoio/Diario/Aula/2002681954/Sebesta_Cap10pp.pdf