Implementação de Linguagens de Programação
Análise de otimizações em algoritmo com GCC

Luis Fernando Rocha de Sousa Alves [1]

[1]Departamento de Informática (DIN) – Universidade estadual de Maringá (UEM)
Av. Colombo, 5790 – UEM – Bloco C56 – Maringá – Paraná – Brazil

ra67403@uem.br

Abstract. *Optimization is a crucial step in the development context of algorithms, where depending on the purpose, different levels of optimization can be applied. Thus, the Network Dijkstra algorithm has been choosen in order to perform the compilation and execution with some levels of optimization from the GCC, measuring its execution time, number of cycles and instructions. In the present work, it is also discussed how the front-end and middle-end analyzes are performed in GCC.*

Resumo. *Otimização é uma etapa crucial no contexto de desenvolvimento de algoritmos, onde dependendo da finalidade, diferentes níveis de otimização podem ser aplicados. Assim, foi selecionado o algoritmo Network Dijkstra, a fim de realizar a compilação e execução com diversos níveis de otimização provindos do GCC, mensurando seu tempo de execução, quantidade de ciclos e instruções. No presente trabalho, é abordado também como as análises front-end e middle-end são realizadas no GCC.*

1. Introdução

Programas, que não são implementados diretamente em linguagem de máquina, necessitam de uma tradução para que possam ser executados, essa tradução pode ser feita por meio de um compilador, sendo este definido como um programa que recebe como entrada um código fonte, e o traduz para uma outra linguagem equivalente [Louden 2004].

Um compilador comum é dividido em dois conjuntos, o conjunto front-end e o conjunto back-end, sendo que o primeiro realiza as análises léxicas, sintáticas e semânticas, e o segundo faz a geração de código intermediário, seleção de instruções e alocação de registradores [AHO et al.]. Desta forma, alguns compiladores, como o GCC[1], tentam otimizar o código gerado de acordo com a finalidade, seja ela tempo de execução, tempo de compilação, tamanho do código, consumo de energia, dentre outros.

Algumas ferramentas, como o time e o perf, podem ser usadas para mensurar o tempo de execução, a quantidade de ciclos e a quantidade de instruções, onde é possível analisar os resultados e tirar conclusões de forma empírica.

2. Arquitetura

- Processador: Intel(R) Core(TM) i5-4460 @ 3.20GHz, 3201 Mhz, 4 Núcleos(s), 4 Processadores lógicos
- Memória RAM: 16GB;
- Armazenamento: SSD 240GB KINGSTON SV300S37A240G

[1]https://gcc.gnu.org

2.1. Software

- Sistema Operacional: Ubuntu 16.04.3 LTS - 64bits;
- GCC: Versão 6.3.0;
- Perf: Versão 4.10.17

3. Análise

Para realizar a análise de desempenho das otimizações, foi utilizado o algoritmo Network Dijkstra, retirado da base de testes cBench[2], para 20 diferentes entradas. O algoritmo é bem conhecido na disciplina de Algoritmo em Grafos, onde a partir de um vértice, são encontradas as menores distâncias para todos os outros vértices alcançáveis, tendo a complexidade de $O(n^2)$, sendo n a quantidade de vértices do grafo de entrada [Barros et al. 2007].

3.1. Otimizações

Cada cenário pode requerer características diferentes, como por exemplo sistemas embarcados precisam minimizar o consumo de energia e de armazenamento, sistemas em tempo real requerem a maximização da velocidade de execução, ou ainda, cenários que precisam compilar código em tempo de execução necessitam de uma compilação mais ágil. As otimizações realizadas no momento da compilação podem suprir parte dessas necessidades, se aplicadas de maneira correta.

O GCC permite habilitar níveis de otimização -O, -O0, -O1, -O2, -O3, -Ofast, -Og e -Os, juntamente com com parâmetros para habilitar ou desabilitar otimizações em específico.

3.1.1. Nível de otimização -O0

O nível de otimização -O0 é o parâmetro padrão, gerando código não otimizado, portanto seu tempo de compilação é o melhor dentre os parâmetros.
De maneira geral, esta otimização apresentou o maior tempo de execução, a maior quantidade de ciclos, maior quantidade de instruções e a menor quantidade de instruções realizadas em um ciclo, apresentando assim o pior desempenho dentre todos os níveis de otimizações. Na medida que o tamanho da entrada aumenta, essas diferenças passam a ser mais perceptíveis.

3.1.2. Níveis de otimização -O e -O1

Os níveis de otimização -O e -O1 são equivalentes, onde busca-se reduzir tamanho de código gerado e melhorar o tempo de execução. Apesar destes benefícios, estes parâmetros de exigem um gasto maior de memória e de tempo de compilação, se comparado ao -O0, porém com o cuidado em relação ao tempo de compilação.
Este nível de otimização apresentou resultados melhores que o conjunto de otimizações em -O0 em todos os testes, tendo reduzido, em quase todos os casos em mais da metade, o tempo de execução, quantidades de ciclos, quantidade de instruções e aumentado a quantidade de instruções realizadas em um ciclo.

[2]http://cTuning.org/cbench

As otimizações habilitadas em -O e -O1 são:
-fauto-inc-dec ,-fbranch-count-reg ,-fcombine-stack-adjustments ,-fcompare-elim ,-fcprop-registers ,-fdce ,-fdefer-pop ,-fdelayed-branch ,-fdse ,-fforward-propagate ,-fguess-branch-probability ,-fif-conversion2 ,-fif-conversion ,-finline-functions-called-once ,-fipa-pure-const ,-fipa-profile ,-fipa-reference ,-fmerge-constants ,-fmove-loop-invariants ,-freorder-blocks ,-fshrink-wrap ,-fshrink-wrap-separate ,-fsplit-wide-types ,-fssa-backprop ,-fssa-phiopt ,-ftree-bit-ccp ,-ftree-ccp ,-ftree-ch ,-ftree-coalesce-vars ,-ftree-copy-prop ,-ftree-dce ,-ftree-dominator-opts ,-ftree-dse ,-ftree-forwprop ,-ftree-fre ,-ftree-phiprop ,-ftree-sink ,-ftree-slsr ,-ftree-sra ,-ftree-pta ,-ftree-ter ,-funit-at-a-time.

3.1.3. Nível de otimização -O2

O nível de otimização -O2 contém todas as otimizações de -O1, buscando melhorar o tempo de execução por meio de todas as otimizações que não envolvem troca de espaço-velocidade.
De maneira geral, esta otimização apresentou resultados melhores que o conjunto de otimizações em O1, porém em alguns casos estes resultados são praticamente imperceptíveis, desta forma, houveram ligeiras reduções de tempo de execução, quantidade de ciclos, quantidade de instruções, porém esta otimização obteve a maior quantidade de instruções realizadas em um ciclo de todas as otimizações.
As otimizações habilitadas em -O2 são as mesmas de -O1, juntamente com:
-fthread-jumps ,-falign-functions -falign-jumps ,-falign-loops -falign-labels ,-fcaller-saves ,-fcrossjumping ,-fcse-follow-jumps -fcse-skip-blocks ,-fdelete-null-pointer-checks ,-fdevirtualize -fdevirtualize-speculatively ,-fexpensive-optimizations ,-fgcse -fgcse-lm ,-fhoist-adjacent-loads ,-finline-small-functions ,-findirect-inlining ,-fipa-cp ,-fipa-bit-cp ,-fipa-vrp ,-fipa-sra ,-fipa-icf ,-fisolate-erroneous-paths-dereference ,-flra-remat ,-foptimize-sibling-calls ,-foptimize-strlen ,-fpartial-inlining ,-fpeephole2 ,-freorder-blocks-algorithm=stc ,-freorder-blocks-and-partition -freorder-functions ,-frerun-cse-after-loop ,-fsched-interblock -fsched-spec ,-fschedule-insns -fschedule-insns2 ,-fstore-merging ,-fstrict-aliasing ,-ftree-builtin-call-dce ,-ftree-switch-conversion -ftree-tail-merge ,-fcode-hoisting ,-ftree-pre ,-ftree-vrp e -fipa-ra.

3.1.4. Nível de otimização -O3

O nível de otimização -O3 contém todas as otimizações de -O2, sendo o nível que possui os melhores resultados em relação ao tempo de execução dentre todos os níveis de otimização, que mantém a integridade e assertividade.
Este nível de otimização apresentou os melhores resultados dentre os níveis -O0, -O1, -O2 e -O3, tendo o menor tempo de execução, menor quantidade de ciclos, menor quantidade de instruções, porém, no quesito de quantidade de instruções realizadas em um ciclo, este nível teve um desempenho pior que o nível -O2.
As otimizações habilitadas em -O3 são as mesmas de -O2, juntamente com:
-finline-functions ,-funswitch-loops ,-fpredictive-commoning ,-fgcse-after-reload ,-ftree-loop-vectorize ,-ftree-loop-distribution ,-ftree-loop-distribute-patterns ,-fsplit-paths ,-ftree-slp-vectorize ,-fvect-cost-model ,-ftree-partial-pre ,-fpeel-loops e -fipa-cp-clone

3.1.5. Nível de otimização -Ofast

O nível de otimização -Ofast está sendo apresentado apenas à título de curiosidade pois não foi utilizado nos testes, apesar deste poder possuir o melhor desempenho no quesito tempo de execução, são habilitadas as otimizações -ffast-math e -fno-protect-parens, que resultam em cálculos matemáticos rápidos, porém, não precisos.
As otimizações habilitadas no nível -Ofast são as mesmas de -O3, juntamente com:
-ffast-math e -fno-protect-parens

3.1.6. Nível de otimização -O1 com -falign-functions

Esta otimização alinha funções ao próximo limite de 2^n, onde n é definido pela máquina quando não especificado. Esta otimização é ativa nos níveis -O2 e -O3.
Esta otimização apresentou resultados semelhantes ao nível -O1, com tempo de execução, quantidade de ciclos quantidade de instruções ligeiramente maiores, com a quantidade de instruções realizadas em um ciclo um pouco menor.

3.1.7. Nível de otimização -O1 com -fgcse

Esta otimização elimina subexpressões comuns globais, juntamente com a execução de constantes globais. Esta otimização é ativa nos níveis -O2, -O3 e -Os.
Esta otimização apresentou resultados semelhantes ao -O1, com tempo de execução, quantidade de ciclos quantidade de instruções ligeiramente maiores, o que diferenciou -fgcse das outras otimizações individuais foi a maior quantidade de instruções realizadas em um ciclo de todas as otimizações individuais em -O1.

3.1.8. Nível de otimização -O3 com -fno-gcse

Este parâmetro desabilita a otimização -fgcse, deixando de eliminar subexpressões comuns globais. Esta otimização pode desabilitar a otimização -fgcse nos níveis -O2, -O3 e -Os.
Esta otimização, dentre todas as otimizações em -O3, apresentou o menor tempo de execução, quantidade de ciclos, quantidade de instruções, e uma das que apresentou as maiores quantidades instruções realizadas em um ciclo.

3.1.9. Nível de otimização -O3 com -fno-guess-branch-probability

Este parâmetro desabilita a otimização -fguess-branch-probability, fazendo assim que não haja a adivinhação de probabilidades de ramo por heurísticas. Esta otimização pode desabilitar a otimização de -fguess-branch-probability nos níveis -O, -O2, -O3 e -Os.
Esta otimização, dentre todas as otimizações em -O3, apresentou tempo execução, quantidade de ciclos e quantidade de instruções ligeiramente maiores que o nível -O3, e com o menor quantidade de instruções e instruções realizadas em um ciclo.

3.1.10. Nível de otimização -O3 com -fno-inline

Esta otimização faz com que o compilador deixe de prestar atenção em palavra-chave “inline”, normalmente usada para evitar que a palavra-chave ”inline”seja expandida. Esta otimização pode desabilitar parte das otimizações em -finline-functions no nível -O3.
Esta otimização, dentre todas as otimizações em -O3, apresentou o maior tempo de execução, quantidade de ciclos, quantidade de instruções e instruções realizadas em um ciclo de todas.

3.1.11. Nível de otimização -O3 com -fno-peephole2

Este parâmetro desabilita a otimização -fpeephole2, assim desabilitando otimizações do tipo peephole, que seriam substituições em conjuntos de instruções que poderiam ser mais breves ou mais rápidas. Esta otimização pode desabilitar a otimização -fpeephole2 nos níveis -O2, -O3 e -Os.
Esta otimização, dentre todas as otimizações em -O3, apresentou tempo execução e quantidade de ciclos ligeiramente maiores, algo quase imperceptível, tendo também praticamente nenhuma variação na quantidade de instruções e quantidade de instruções realizadas em um ciclo.

3.1.12. Otimização -fno-toplevel-reorder

Esta otimização faz com que o compilador deixe de reordenar funções de alto nível, variáveis e alguns tipos de declarações, fazendo com que variáveis estáticas não referenciadas não sejam removidas. Esta otimização pode ser habilitada nos níveis -O0, -O1, -O2 e -O3. Esta otimização, dentre todas as otimizações em -O3, apresentou tempo execução e quantidade de ciclos ligeiramente maiores, algo quase imperceptível, tendo também praticamente nenhuma variação na quantidade de instruções e quantidade de instruções realizadas em um ciclo.

3.1.13. Nível de otimização -O3 com -fno-zero-initialized-in-bss

Este parâmetro desabilita a otimização -fzero-initialized-in-bss, desta forma, mesmo se o houver suporte à sessão BSS, variáveis inicializadas com zero, não iriam para esta sessão. Esta otimização pode desabilitar a otimização -fzero-initialized-in-bss nos níveis -O0, -O1, -O2 e -O3.
Esta otimização, dentre todas as otimizações em -O3, apresentou tempo execução e quantidade de ciclos ligeiramente maiores, algo quase imperceptível, tendo também praticamente nenhuma variação na quantidade de instruções e quantidade de instruções realizadas em um ciclo.

3.1.14. Nível de otimização -O1 com -fpeel-loops

Este parâmetro de otimização realiza a remoção completa de laços de repetição, que tenham informações suficientes e com pequenos números de constantes de iteração, sendo

ativa no nível -O3, ou pelo parâmetro -fprofile-use.
Esta otimização, dentre as aplicados em -O1, foi o que menos apresentou mudanças, obtendo poucas mudanças no tempo de execução, quantidade de ciclos, quantidade de instruções e instruções realizadas em um ciclo.

3.1.15. Nível de otimização -O1 com-fpeephole2

Este parâmetro habilita otimizações do tipo peephole, onde conjuntos de instruções que poderiam ser mais breves ou mais rápidas são substituídas por trechos equivalentes. Esta otimização pode ser habilitada nos níveis -O2, -O3 e -Os.
Esta otimização, dentre as aplicadas em -O1, foi a que apresentou o tempo de execução, quantidade de ciclos, e um leve aumento na quantidade de instruções realizadas em um ciclo. A quantidade de instruções não apresentou variação relevante.

3.2. Resultados

Foram usados 20 arquivos diferentes de entrada, retirados da base de teste cBench, sendo que seu tamanho e complexidade estão em ordem crescente, onde a entrada 1.dat é a menor e mais simples, e a entrada 20.dat, a maior e mais complexa.

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,001	936.671	704.191	0,75
-O1	0,001	941.752	696.216	0,74
-O1 -falign-functions	0,001	942.107	697.898	0,74
-O1 -fgcse	0,001	929.435	698.933	0,75
-O1 -fpeel-loops	0,001	926.520	693.953	0,75
-O1 -fpeephole2	0,001	956.948	696.694	0,73
-O2	0,001	937.251	693.389	0,74
-O3	0,001	946.829	693.937	0,73
-O3 -fno-gcse	0,001	930.094	690.848	0,74
-O3 -fno-guess-branch-probability	0,001	948.624	695.277	0,73
-O3 -fno-inline	0,001	955.365	700.595	0,74
-O3 -fno-peephole2	0,001	939.765	692.298	0,74
-O3 -fno-toplevel-reorder	0,001	934.369	692.611	0,74
-O3 -fno-zero-initialized-in-bss	0,001	945.065	694.480	0,74

Tabela 1. Média das execuções do algoritmo com a entrada 1.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,002	2.869.712	3.664.676	1,28
-O1	0,002	2.153.493	2.833.809	1,32
-O1 -falign-functions	0,002	2.162.899	2.838.731	1,31
-O1 -fgcse	0,002	2.159.006	2.875.004	1,33
-O1 -fpeel-loops	0,002	2.173.911	2.841.729	1,31
-O1 -fpeephole2	0,002	2.168.049	2.840.812	1,31
-O2	0,002	2.178.509	2.813.369	1,29
-O3	0,002	2.070.129	2.592.662	1,25
-O3 -fno-gcse	0,002	2.027.052	2.581.938	1,27
-O3 -fno-guess-branch-probability	0,002	2.115.792	2.550.515	1,21
-O3 -fno-inline	0,002	2.155.986	2.783.742	1,29
-O3 -fno-peephole2	0,002	2.090.084	2.594.039	1,24
-O3 -fno-toplevel-reorder	0,002	2.079.085	2.594.937	1,25
-O3 -fno-zero-initialized-in-bss	0,002	2.087.333	2.598.239	1,24

Tabela 2. Média das execuções do algoritmo com a entrada 2.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,011	18.130.567	27.358.789	1,51
-O1	0,007	10.131.285	18.205.824	1,80
-O1 -falign-functions	0,007	10.160.722	18.216.484	1,79
-O1 -fgcse	0,007	10.125.096	18.675.415	1,85
-O1 -fpeel-loops	0,007	10.245.853	18.213.369	1,78
-O1 -fpeephole2	0,006	10.105.667	18.223.551	1,80
-O2	0,007	10.365.162	18.100.738	1,75
-O3	0,006	9.260.224	15.683.264	1,69
-O3 -fno-gcse	0,006	9.130.867	15.690.078	1,72
-O3 -fno-guess-branch-probability	0,007	9.943.316	15.255.500	1,54
-O3 -fno-inline	0,006	10.035.024	17.675.423	1,76
-O3 -fno-peephole2	0,006	9.212.663	15.684.097	1,70
-O3 -fno-toplevel-reorder	0,006	9.259.464	15.686.398	1,69
-O3 -fno-zero-initialized-in-bss	0,006	9.264.564	15.686.859	1,69

Tabela 3. Média das execuções do algoritmo com a entrada 3.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,037	71.081.969	105.857.043	1,49
-O1	0,019	35.861.474	67.208.080	1,88
-O1 -falign-functions	0,020	35.874.008	67.209.061	1,87
-O1 -fgcse	0,019	35.857.286	69.120.524	1,93
-O1 -fpeel-loops	0,019	36.098.288	67.214.553	1,86
-O1 -fpeephole2	0,019	35.810.758	67.246.294	1,88
-O2	0,019	36.869.088	66.914.895	1,82
-O3	0,018	32.555.384	57.085.224	1,75
-O3 -fno-gcse	0,018	31.929.834	57.088.303	1,79
-O3 -fno-guess-branch-probability	0,019	35.599.107	55.269.006	1,55
-O3 -fno-inline	0,019	35.674.907	65.107.957	1,83
-O3 -fno-peephole2	0,017	32.538.278	57.087.347	1,76
-O3 -fno-toplevel-reorder	0,018	32.597.967	57.091.880	1,75
-O3 -fno-zero-initialized-in-bss	0,018	32.611.518	57.091.186	1,75

Tabela 4. Média das execuções do algoritmo com a entrada 4.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,095	186.664.044	273.417.323	1,47
-O1	0,047	90.623.770	170.251.947	1,88
-O1 -falign-functions	0,048	90.691.338	170.260.365	1,88
-O1 -fgcse	0,047	91.087.693	175.314.556	1,92
-O1 -fpeel-loops	0,047	91.152.464	170.266.401	1,87
-O1 -fpeephole2	0,047	90.567.778	170.322.912	1,88
-O2	0,046	87.352.036	169.660.006	1,94
-O3	0,040	77.050.632	143.906.644	1,87
-O3 -fno-gcse	0,040	76.201.037	143.896.861	1,89
-O3 -fno-guess-branch-probability	0,042	79.052.205	139.015.241	1,76
-O3 -fno-inline	0,046	87.411.363	164.823.279	1,89
-O3 -fno-peephole2	0,040	76.939.282	143.894.121	1,87
-O3 -fno-toplevel-reorder	0,040	77.263.698	143.904.876	1,87
-O3 -fno-zero-initialized-in-bss	0,040	77.745.556	143.902.368	1,85

Tabela 5. Média das execuções do algoritmo com a entrada 5.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,202	399.962.666	578.702.928	1,45
-O1	0,098	190.964.334	356.405.025	1,87
-O1 -falign-functions	0,099	191.248.593	356.414.290	1,86
-O1 -fgcse	0,098	190.876.316	367.188.152	1,93
-O1 -fpeel-loops	0,098	191.884.731	356.402.632	1,86
-O1 -fpeephole2	0,098	190.672.531	356.509.663	1,87
-O2	0,094	183.618.654	355.386.685	1,94
-O3	0,084	162.515.464	300.657.500	1,85
-O3 -fno-gcse	0,083	159.423.751	300.654.188	1,89
-O3 -fno-guess-branch-probability	0,086	166.285.133	290.185.249	1,75
-O3 -fno-inline	0,095	183.699.432	344.997.321	1,88
-O3 -fno-peephole2	0,084	162.511.852	300.659.971	1,85
-O3 -fno-toplevel-reorder	0,083	162.132.947	300.656.102	1,86
-O3 -fno-zero-initialized-in-bss	0,084	163.118.260	300.661.865	1,84

Tabela 6. Média das execuções do algoritmo com a entrada 6.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,365	723.596.352	1.033.680.861	1,43
-O1	0,174	342.396.685	632.106.998	1,85
-O1 -falign-functions	0,176	342.466.401	632.132.042	1,85
-O1 -fgcse	0,174	342.169.363	651.354.803	1,90
-O1 -fpeel-loops	0,174	343.355.554	632.107.093	1,84
-O1 -fpeephole2	0,173	341.753.872	632.270.383	1,85
-O2	0,167	329.113.356	630.602.542	1,92
-O3	0,149	292.777.402	533.148.971	1,82
-O3 -fno-gcse	0,145	286.289.638	533.151.678	1,86
-O3 -fno-guess-branch-probability	0,152	298.689.031	514.368.114	1,72
-O3 -fno-inline	0,167	328.682.684	611.922.844	1,86
-O3 -fno-peephole2	0,149	293.201.476	533.151.019	1,82
-O3 -fno-toplevel-reorder	0,149	292.827.851	533.158.313	1,82
-O3 -fno-zero-initialized-in-bss	0,149	293.845.249	533.148.981	1,82

Tabela 7. Média das execuções do algoritmo com a entrada 7.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,629	1.251.652.575	1.775.943.069	1,42
-O1	0,297	587.167.801	1.080.520.592	1,84
-O1 -falign-functions	0,296	587.646.717	1.080.523.883	1,84
-O1 -fgcse	0,298	589.037.642	1.113.671.217	1,89
-O1 -fpeel-loops	0,297	589.345.980	1.080.525.473	1,84
-O1 -fpeephole2	0,296	585.987.353	1.080.758.709	1,85
-O2	0,285	564.117.488	1.078.280.302	1,91
-O3	0,255	504.167.327	910.784.942	1,81
-O3 -fno-gcse	0,247	490.357.367	910.741.912	1,86
-O3 -fno-guess-branch-probability	0,259	512.636.691	878.283.323	1,71
-O3 -fno-inline	0,284	563.186.355	1.045.988.742	1,86
-O3 -fno-peephole2	0,255	505.261.733	910.744.501	1,81
-O3 -fno-toplevel-reorder	0,255	504.776.066	910.765.333	1,81
-O3 -fno-zero-initialized-in-bss	0,256	506.639.264	910.775.259	1,80

Tabela 8. Média das execuções do algoritmo com a entrada 8.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	0,991	1.971.893.605	2.770.108.995	1,41
-O1	0,463	921.058.836	1.679.184.957	1,82
-O1 -falign-functions	0,464	922.107.348	1.679.181.563	1,82
-O1 -fgcse	0,464	921.139.620	1.730.761.558	1,88
-O1 -fpeel-loops	0,464	921.765.561	1.679.174.928	1,82
-O1 -fpeephole2	0,462	918.910.067	1.679.492.168	1,83
-O2	0,445	884.724.206	1.676.151.818	1,90
-O3	0,400	793.902.639	1.415.837.915	1,78
-O3 -fno-gcse	0,388	770.512.884	1.415.807.620	1,84
-O3 -fno-guess-branch-probability	0,406	805.955.637	1.365.171.510	1,70
-O3 -fno-inline	0,443	881.304.616	1.625.710.944	1,84
-O3 -fno-peephole2	0,400	796.538.679	1.415.827.048	1,78
-O3 -fno-toplevel-reorder	0,400	794.266.383	1.415.827.954	1,78
-O3 -fno-zero-initialized-in-bss	0,402	798.697.596	1.415.814.395	1,77

Tabela 9. Média das execuções do algoritmo com a entrada 9.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	1,496	2.980.865.572	4.163.177.355	1,40
-O1	0,697	1.387.527.422	2.516.711.787	1,81
-O1 -falign-functions	0,700	1.391.413.311	2.516.706.223	1,81
-O1 -fgcse	0,699	1.389.131.964	2.594.188.607	1,87
-O1 -fpeel-loops	0,699	1.395.138.881	2.516.691.278	1,80
-O1 -fpeephole2	0,696	1.384.887.189	2.517.135.749	1,82
-O2	0,671	1.332.987.896	2.512.628.646	1,89
-O3	0,603	1.199.375.336	2.121.684.242	1,77
-O3 -fno-gcse	0,583	1.159.816.400	2.121.640.059	1,83
-O3 -fno-guess-branch-probability	0,610	1.216.557.195	2.045.396.918	1,68
-O3 -fno-inline	0,667	1.327.321.077	2.436.653.818	1,84
-O3 -fno-peephole2	0,602	1.207.562.362	2.121.683.308	1,76
-O3 -fno-toplevel-reorder	0,602	1.200.271.059	2.121.650.543	1,77
-O3 -fno-zero-initialized-in-bss	0,605	1.205.212.852	2.121.702.624	1,76

Tabela 10. Média das execuções do algoritmo com a entrada 10.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	2,123	4.236.766.142	5.857.836.129	1,38
-O1	0,986	1.965.972.616	3.532.926.822	1,80
-O1 -falign-functions	0,990	1.972.792.160	3.532.950.313	1,79
-O1 -fgcse	0,991	1.969.889.574	3.641.643.147	1,85
-O1 -fpeel-loops	0,989	1.976.004.206	3.532.975.341	1,79
-O1 -fpeephole2	0,985	1.963.917.629	3.533.487.062	1,80
-O2	0,949	1.890.594.130	3.527.808.875	1,87
-O3	0,857	1.706.000.088	2.980.168.330	1,75
-O3 -fno-gcse	0,826	1.645.698.091	2.980.117.357	1,81
-O3 -fno-guess-branch-probability	0,867	1.726.544.773	2.873.015.620	1,66
-O3 -fno-inline	0,946	1.882.045.966	3.421.091.711	1,82
-O3 -fno-peephole2	0,858	1.709.925.686	2.980.169.194	1,74
-O3 -fno-toplevel-reorder	0,856	1.707.285.856	2.980.172.985	1,75
-O3 -fno-zero-initialized-in-bss	0,861	1.714.311.068	2.980.159.784	1,74

Tabela 11. Média das execuções do algoritmo com a entrada 11.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	3,004	6.002.347.270	8.201.681.638	1,37
-O1	1,402	2.796.592.344	4.936.876.938	1,77
-O1 -falign-functions	1,409	2.808.546.619	4.936.859.729	1,76
-O1 -fgcse	1,407	2.806.756.980	5.088.539.028	1,82
-O1 -fpeel-loops	1,408	2.816.531.655	4.936.899.995	1,75
-O1 -fpeephole2	1,402	2.782.810.021	4.937.508.374	1,78
-O2	1,349	2.682.646.615	4.930.310.628	1,84
-O3	1,216	2.430.020.393	4.166.685.512	1,72
-O3 -fno-gcse	1,176	2.339.965.579	4.166.608.144	1,78
-O3 -fno-guess-branch-probability	1,237	2.456.301.057	4.016.966.155	1,64
-O3 -fno-inline	1,343	2.671.272.543	4.781.163.190	1,79
-O3 -fno-peephole2	1,220	2.428.430.416	4.166.648.494	1,72
-O3 -fno-toplevel-reorder	1,220	2.428.988.977	4.166.678.048	1,72
-O3 -fno-zero-initialized-in-bss	1,224	2.437.407.142	4.166.732.849	1,71

Tabela 12. Média das execuções do algoritmo com a entrada 12.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	4,063	8.118.030.890	10.846.092.676	1,34
-O1	1,926	3.833.498.571	6.518.126.668	1,70
-O1 -falign-functions	1,933	3.845.997.360	6.518.118.485	1,70
-O1 -fgcse	1,927	3.843.849.781	6.717.782.693	1,75
-O1 -fpeel-loops	1,927	3.846.559.494	6.518.122.843	1,70
-O1 -fpeephole2	1,903	3.796.056.313	6.518.907.492	1,72
-O2	1,849	3.656.114.709	6.510.239.218	1,78
-O3	1,679	3.327.259.742	5.505.512.652	1,66
-O3 -fno-gcse	1,610	3.215.821.346	5.505.441.963	1,71
-O3 -fno-guess-branch-probability	1,684	3.357.867.714	5.308.214.558	1,58
-O3 -fno-inline	1,844	3.645.474.409	6.313.786.891	1,73
-O3 -fno-peephole2	1,673	3.325.604.229	5.505.549.269	1,66
-O3 -fno-toplevel-reorder	1,669	3.328.412.670	5.505.583.289	1,65
-O3 -fno-zero-initialized-in-bss	1,675	3.346.369.934	5.505.525.215	1,65

Tabela 13. Média das execuções do algoritmo com a entrada 13.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	5,393	10.769.924.144	14.160.903.284	1,32
-O1	2,571	5.137.767.097	8.500.259.716	1,66
-O1 -falign-functions	2,584	5.154.936.168	8.500.336.815	1,65
-O1 -fgcse	2,579	5.149.478.443	8.760.997.835	1,70
-O1 -fpeel-loops	2,582	5.146.599.096	8.500.267.482	1,65
-O1 -fpeephole2	2,542	5.076.417.329	8.501.196.924	1,68
-O2	2,453	4.895.513.036	8.490.753.526	1,74
-O3	2,235	4.468.982.150	7.179.441.844	1,61
-O3 -fno-gcse	2,161	4.313.775.927	7.179.357.827	1,67
-O3 -fno-guess-branch-probability	2,260	4.508.729.244	6.921.656.044	1,54
-O3 -fno-inline	2,444	4.877.424.885	8.233.682.415	1,69
-O3 -fno-peephole2	2,238	4.470.274.314	7.179.474.902	1,61
-O3 -fno-toplevel-reorder	2,240	4.465.427.350	7.179.520.776	1,61
-O3 -fno-zero-initialized-in-bss	2,250	4.486.913.866	7.179.500.554	1,60

Tabela 14. Média das execuções do algoritmo com a entrada 14.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	7,096	14.186.110.695	18.038.579.135	1,27
-O1	3,456	6.911.983.993	10.816.311.872	1,57
-O1 -falign-functions	3,473	6.938.579.684	10.816.373.153	1,56
-O1 -fgcse	3,467	6.918.621.506	11.147.960.740	1,61
-O1 -fpeel-loops	3,462	6.923.577.751	10.816.366.649	1,56
-O1 -fpeephole2	3,408	6.805.980.992	10.817.452.057	1,59
-O2	3,292	6.576.712.513	10.805.159.851	1,64
-O3	3,021	6.027.545.961	9.138.219.258	1,52
-O3 -fno-gcse	2,921	5.826.850.623	9.137.980.789	1,57
-O3 -fno-guess-branch-probability	3,046	6.084.346.498	8.809.938.609	1,45
-O3 -fno-inline	3,281	6.557.973.945	10.477.922.048	1,60
-O3 -fno-peephole2	3,020	6.039.021.814	9.138.095.403	1,52
-O3 -fno-toplevel-reorder	3,019	6.028.874.874	9.138.167.535	1,52
-O3 -fno-zero-initialized-in-bss	3,037	6.056.317.695	9.138.107.483	1,51

Tabela 15. Média das execuções do algoritmo com a entrada 15.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	9,115	18.229.680.388	22.378.834.107	1,23
-O1	4,526	9.047.186.662	13.405.369.312	1,48
-O1 -falign-functions	4,548	9.087.647.404	13.405.365.063	1,48
-O1 -fgcse	4,534	9.044.980.119	13.815.216.508	1,53
-O1 -fpeel-loops	4,531	9.060.187.504	13.405.502.403	1,48
-O1 -fpeephole2	4,454	8.897.505.755	13.406.592.130	1,51
-O2	4,310	8.611.121.308	13.392.305.336	1,56
-O3	3,976	7.924.943.669	11.332.493.898	1,43
-O3 -fno-gcse	3,849	7.698.747.518	11.332.569.724	1,47
-O3 -fno-guess-branch-probability	4,011	8.008.001.853	10.926.685.001	1,36
-O3 -fno-inline	4,297	8.585.699.448	12.987.471.296	1,51
-O3 -fno-peephole2	3,978	7.936.968.661	11.332.545.894	1,43
-O3 -fno-toplevel-reorder	3,971	7.935.826.583	11.332.584.541	1,43
-O3 -fno-zero-initialized-in-bss	3,989	7.967.402.512	11.332.535.182	1,42

Tabela 16. Média das execuções do algoritmo com a entrada 16.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	11,865	23.725.620.223	28.041.233.067	1,18
-O1	5,994	11.983.181.525	16.780.965.085	1,40
-O1 -falign-functions	6,023	12.041.789.734	16.780.779.416	1,39
-O1 -fgcse	5,995	11.986.174.468	17.291.827.267	1,45
-O1 -fpeel-loops	6,004	12.003.666.194	16.780.925.701	1,40
-O1 -fpeephole2	5,913	11.792.529.869	16.782.163.507	1,42
-O2	5,717	11.431.230.512	16.765.493.652	1,47
-O3	5,295	10.588.844.448	14.198.261.069	1,34
-O3 -fno-gcse	5,161	10.273.860.521	14.198.200.784	1,38
-O3 -fno-guess-branch-probability	5,341	10.680.772.561	13.691.659.252	1,28
-O3 -fno-inline	5,704	11.390.920.636	16.260.320.684	1,43
-O3 -fno-peephole2	5,295	10.582.892.179	14.198.107.578	1,34
-O3 -fno-toplevel-reorder	5,291	10.579.797.379	14.198.023.052	1,34
-O3 -fno-zero-initialized-in-bss	5,316	10.631.808.064	14.198.111.968	1,34

Tabela 17. Média das execuções do algoritmo com a entrada 17.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	14,799	29.591.054.192	34.390.539.375	1,16
-O1	7,539	15.052.953.867	20.570.799.939	1,37
-O1 -falign-functions	7,580	15.148.018.881	20.570.786.380	1,36
-O1 -fgcse	7,549	15.079.979.979	21.200.182.629	1,41
-O1 -fpeel-loops	7,556	15.091.707.401	20.570.777.958	1,36
-O1 -fpeephole2	7,414	14.826.902.271	20.572.512.728	1,39
-O2	7,197	14.394.859.624	20.553.138.688	1,43
-O3	6,668	13.320.009.176	17.391.592.404	1,31
-O3 -fno-gcse	6,495	12.989.841.254	17.391.512.550	1,34
-O3 -fno-guess-branch-probability	6,735	13.462.671.934	16.767.514.831	1,25
-O3 -fno-inline	7,177	14.342.017.921	19.930.403.156	1,39
-O3 -fno-peephole2	6,675	13.363.075.156	17.391.653.283	1,30
-O3 -fno-toplevel-reorder	6,669	13.320.449.538	17.391.612.766	1,30
-O3 -fno-zero-initialized-in-bss	6,698	13.375.115.598	17.391.699.619	1,30

Tabela 18. Média das execuções do algoritmo com a entrada 18.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	18,504	36.997.026.657	41.605.597.256	1,12
-O1	9,580	19.131.832.571	24.865.844.798	1,30
-O1 -falign-functions	9,622	19.241.522.536	24.865.988.411	1,29
-O1 -fgcse	9,586	19.147.066.241	25.623.122.112	1,34
-O1 -fpeel-loops	9,582	19.128.069.202	24.865.829.526	1,30
-O1 -fpeephole2	9,405	18.807.238.118	24.867.911.577	1,32
-O2	9,138	18.273.215.888	24.845.751.685	1,36
-O3	8,496	16.983.273.369	21.043.116.408	1,24
-O3 -fno-gcse	8,289	16.570.463.209	21.043.008.200	1,27
-O3 -fno-guess-branch-probability	8,585	17.162.219.374	20.292.360.342	1,18
-O3 -fno-inline	9,107	18.204.130.337	24.096.197.334	1,32
-O3 -fno-peephole2	8,502	17.017.483.701	21.043.172.278	1,24
-O3 -fno-toplevel-reorder	8,504	16.996.788.530	21.043.158.425	1,24
-O3 -fno-zero-initialized-in-bss	8,529	17.061.456.314	21.043.147.165	1,23

Tabela 19. Média das execuções do algoritmo com a entrada 19.dat

Otimização	Tempo (seg)	Ciclos	Instruções	Ins/Cic
-O0	22,375	44.733.602.244	49.758.286.930	1,11
-O1	11,604	23.240.199.299	29.722.073.231	1,28
-O1 -falign-functions	11,688	23.348.765.465	29.722.194.565	1,27
-O1 -fgcse	11,641	23.237.722.434	30.626.757.072	1,32
-O1 -fpeel-loops	11,630	23.265.807.593	29.722.092.663	1,28
-O1 -fpeephole2	11,421	22.834.024.295	29.724.172.348	1,30
-O2	11,101	22.194.805.502	29.699.226.625	1,34
-O3	10,339	20.657.191.867	25.156.598.241	1,22
-O3 -fno-gcse	10,092	20.189.035.721	25.156.669.126	1,25
-O3 -fno-guess-branch-probability	10,452	20.894.640.253	24.258.804.476	1,16
-O3 -fno-inline	11,073	22.140.326.418	28.803.114.951	1,30
-O3 -fno-peephole2	10,354	20.674.035.015	25.156.796.143	1,22
-O3 -fno-toplevel-reorder	10,343	20.681.639.748	25.156.705.293	1,22
-O3 -fno-zero-initialized-in-bss	10,378	20.731.971.903	25.156.786.689	1,21

Tabela 20. Média das execuções do algoritmo com a entrada 20.dat

4. GCC

GCC é o nome dado à Coleção de Compiladores GNU, onde é possível compilar uma série de linguagens como C, C++, Objective-C, Objective-C++, Java, Fortran, Ada e Go. Da mesma forma, o nome GCC também é usado para se referir ao sistema de compilação como um todo, e mais especificamente para a fase independente de linguagem do compilador, como por exemplo as otimizações. [Stallman et al. 1999]
Junto com os módulos do GCC, tem-se o CPP[3], sendo o pré processador da linguagem C, o qual realiza uma série de transformações na linguagem de entrada, as quais deveriam seguir uma rígida ordem sequencial, porém, o CPP as realiza de uma única vez por motivos de desempenho [GNU_Compiler_Collection 2017]

4.1. Análise léxica

Segundo a documentação [GNU_Compiler_Collection 2017], o arquivo de entrada é lido, ignorando caracteres sem relevância, e separado em linhas, onde posteriormente serão divididos em tokens.

```
const struct c_common_resword c_common_reswords[] = {
  ...
  { "for",      RID_FOR, 0},
  { "friend",   RID_FRIEND, D_CXXONLY | D_CXXWARN},
  { "goto",     RID_GOTO, 0},
  { "if",       RID_IF, 0},
  { "inline",   RID_INLINE, D_EXT89},
  { "int",      RID_INT, 0},
  ...
};
```

Exte exemplo é uma estrutura escrita na linguagem C, a qual é percorrida por um laço de repetição onde os tokens de palavras reservadas são reconhecidos [GCC_Build_6.3.0 2016].

4.2. Análise sintática

A partir dos tokens gerados pela análise léxica, etapa de análise sintática verifica se a sequência de tokens fazem parte da linguagem gerada pela gramática [AHO et al.].
Atualmente o GCC utiliza um analisador sintático descendente recursivo LR[4] onde a entrada é lida da esquerda para a direita, produzindo uma derivação mais à direita, utilizando a técnica look-ahead, que determina o caminho percorrido pelo analisador com base no próximo token. Este analisador substituiu o antigo Bison[5] afim de ganhar manutenabilidade e performance, o que auxiliou na implementação do OpenMP[6] e outras extensões [Myers 2004].

[3]https://gcc.gnu.org/onlinedocs/gcc-6.3.0/cpp/
[4]Left-Right
[5]https://www.gnu.org/software/bison/
[6]http://www.openmp.org/

```
static void
c_parser_statement_after_labels (c_parser *parser, bool *if_p, vec<tree
    > *chain) {
  ...
  switch (c_parser_peek_token (parser)->type) {
  ...
  case RID_IF:
     c_parser_if_statement (parser, if_p, chain); break;
  case RID_SWITCH:
        c_parser_switch_statement (parser); break;
  case RID_WHILE:
        c_parser_while_statement (parser, false, if_p); break;
  case RID_DO:
        c_parser_do_statement (parser, false); break;
  case RID_FOR:
        c_parser_for_statement (parser, false, if_p); break;
  ...
}

static void
c_parser_while_statement (c_parser *parser, bool ivdep, bool *if_p) {
  tree block, cond, body, save_break, save_cont;
  location_t loc;
  gcc_assert (c_parser_next_token_is_keyword (parser, RID_WHILE));
  token_indent_info while_tinfo = get_token_indent_info (
     c_parser_peek_token (parser));
  c_parser_consume_token (parser);
  block = c_begin_compound_stmt (flag_isoc99);
  loc = c_parser_peek_token (parser)->location;
  cond = c_parser_paren_condition (parser);
  ...
}
static tree
c_parser_paren_condition (c_parser *parser)
{
  tree cond;
  if (!c_parser_require (parser, CPP_OPEN_PAREN, "expected %<(%>"))
    return error_mark_node;
  cond = c_parser_condition (parser);
  c_parser_skip_until_found (parser, CPP_CLOSE_PAREN, "expected %<)%>")
     ;
  return cond;
}
```

Este exemplo de código utiliza o tipo do próximo token para decidir qual caminho do algoritmo seguir, como por exemplo na função *c_parser_statement_after_labels*, caso o token for do tipo RID_WHILE, então o próximo passo é tratar os tokens que devem construir o laço de repetição while, onde a condição e corpo são sintaticamente analisados [GCC_Build_6.3.0 2016]. Na função *c_parser_while_statement* as etapas da estrutura do laço de repetição while são analisadas com base nos próximos tokens, conforme o código acima. Uma dessas etapas é a análise sintática da condição do laço, realizada pela função *c_parser_paren_condition*, a qual faz as verificações e valida a condição.
Desta forma a árvore sintática é preenchida, estando assim apta a ser analisada semanticamente [GCC_Build_6.3.0 2016].

4.3. Análise semântica

A fase de análise semântica de um compilador conecta definições de variáveis à seus usos, juntamente com a validação das tipagens de expressões, definições de escopo e traduz a sintaxe abstrata em uma representação mais simples para que possa ser gerado código de máquina [Appel 2004].

```
static void
c_parser_while_statement (c_parser *parser, bool ivdep, bool *if_p)
{
...
  if(ivdep && cond != error_mark_node)
    cond = build2 (ANNOTATE_EXPR, TREE_TYPE (cond), cond, build_int_cst
       (integer_type_node, annot_expr_ivdep_kind));
  save_break = c_break_label;
  c_break_label = NULL_TREE;
  save_cont = c_cont_label;
  c_cont_label = NULL_TREE;

  token_indent_info body_tinfo = get_token_indent_info (
     c_parser_peek_token (parser));

  body = c_parser_c99_block_statement (parser, if_p);
  c_finish_loop (loc, cond, NULL, body, c_break_label, c_cont_label,
     true);
  add_stmt (c_end_compound_stmt (loc, block, flag_isoc99));

  token_indent_info next_tinfo
    = get_token_indent_info (c_parser_peek_token (parser));
  warn_for_misleading_indentation (while_tinfo, body_tinfo, next_tinfo)
     ;

  c_break_label = save_break;
  c_cont_label = save_cont;
}
int comptypes (tree type1, tree type2) {
  const struct tagged_tu_seen_cache
  * tagged_tu_seen_base1 = tagged_tu_seen_base;
  int val;
  val = comptypes_internal (type1, type2, NULL, NULL);
  free_all_tagged_tu_seen_up_to (tagged_tu_seen_base1);
  return val;
}
```

No GCC, a análise semântica ocorre juntamente com a sintática, para ganho de performance, como pode ser visto na mesma função *c_parser_while_statement*, a qual foi utilizada para exemplificar a análise sintática. Já a função *comptypes* realiza a verificação se dois tipos são compatíveis. [GCC_Build_6.3.0 2016]

4.4. Código intermediário

Após a análise semântica do compilador, já seria possível traduzir a sintaxe abstrata para código de máquina, porém esta tradução é feita, antes, para um código intermediário, a fim de trazer portabilidade e manutenibilidade. Uma representação abstrata pode representar as operações sem se comprometer com detalhes específicos da máquina e da

linguagem original [Appel 2004]. No GCC, após o front-end, e antes do back-end, no middle-end encontra-se o GIMPLE, que é uma família de representações intermediárias baseadas na estrutura de dados da árvore, derivado de GENERIC e bastante influenciado por SIMPLE IL, usado pelo projeto compilador McCAT. Toda estrutura de controle utilizada no GENERIC são reduzidas em saltos condicionais, os limites léxicos são removidos, e também, as regiões de exceção são convertidas em uma região de exceção lateral. A etapa de conversão de GENERIC para GIMPLE é chamada de gimplifier, onde de maneira recursiva, são geradas tuplas GIMPLE fora das extensões originais de GENERIC [GCC_WIKI 2008].

5. Conclusões

Com os testes realizados, e a explicação das etapas realizadas no front-end e middle-end no GCC, é possível concluir que o desempenho do algoritmo está diretamente relacionado às otimizações aplicadas nele, como foi possível observar do nível -O0 até o nível -O3, o tempo de execução, quantidade de instruções e quantidade de ciclos melhoravam a cada conjunto de otimizações adicionadas. Foi possível concluir também que mesmo otimizações unitárias, sendo habilitadas ou desabilitadas em níveis de otimização, podem apresentar impactos tão relevantes quanto um conjunto de otimizações completo, como foi possível ver nas análises dos testes.
Por fim, foi possível compreender melhor o funcionamento do GCC, que segue um padrão de front-end bem conhecido, com suas particularidades, mas de fácil entendimento.

Referências

[AHO et al.] AHO, A., SETHI, R., and LAM, S. Compiladores: princípios, técnicas e ferramentas.[sl]: Longman do brasil, 2008. *ISBN*, 358171665:1.

[Appel 2004] Appel, A. W. (2004). *Modern compiler implementation in C*. Cambridge university press.

[Barros et al. 2007] Barros, E. A., Pamboukian, S. V., and Zamboni, L. C. (2007). Algoritmo de dijkstra: apoio didático e multidisciplinar na implementação, simulação e utilização computacional. In *International Conference on Engineering and Computer Education*.

[GCC_Build_6.3.0 2016] GCC_Build_6.3.0 (2016). Gcc 6.3.0 release.

[GCC_WIKI 2008] GCC_WIKI (2008). Gimple - gcc wiki.

[GNU_Compiler_Collection 2017] GNU_Compiler_Collection (2017). The c preprocessor 6.3.0.

[Louden 2004] Louden, K. C. (2004). *Compiladores-Princípios e Práticas*. Cengage Learning Editores.

[Myers 2004] Myers, J. S. (2004). New c parser [patch].

[Stallman et al. 1999] Stallman, R. M. et al. (1999). *Using and porting the GNU compiler collection*, volume 86. Free Software Foundation.