Trabalho de ARQD1 - Análise e desenvolvimento de Sistemas.

IFSP- Guarulhos.

Professor Robson Ferreira Lopes

Alunos:	Prontuários
Inacio Carvalho de Oliveira	3015076
Heloisa Sousa Chaves	301536X
Lucas Morais de Oliveira	3015106

A relação entre Assembly e C:

As similaridades entre as linguagens em Microcontroladores.

Resumo:

A dialética entre linguagens foi algo que revolucionou a tecnologia contemporânea. Visando encontrar essas relações, o presente trabalho faz uma apresentação breve das relações entre as linguagens Assembly e C, juntamente com um experimento que proporciona uma análise de seus comportamentos. Neste sentido, visamos relacionar as duas, com exemplos de funcionalidades para Micro Controladores Lógicos, com a finalidade de demonstrar suas semelhanças, as diferenças, e o objetivo de cada linguagem de programação para controlá-lo. Trouxemos exemplos com dois microcontroladores PIC família 16F, para alternar o sinal de comutação de uma saída lógica decorrente do sinal de saída de um botão *Pull-up*, fazendo assim que acenda um LED enquanto pressionado. Para melhor ilustrar como funcionam os conceitos de programação para este tipo de dispositivo.

Índice:

Introdução

O ser humano sempre precisou realizar cálculos, sejam eles para armazenar a comida, contar os animais ou construir uma casa. Os primeiros computadores do mundo eram as pessoas. O "computador" era uma profissão cujo trabalho era executar os cálculos repetitivos exigidos para computar tabelas de navegação, cartas de marés e posições planetárias. Devido aos erros decorrentes do cansaço dos "computadores" humanos, durante vários séculos, os inventores procuraram por uma maneira de mecanizar essa tarefa.

Uma das relações primordiais na história da computação e da arquitetura de computadores foi a solução da dialética entre linguagens. Principalmente nas linguagens computacionais de baixo nível, (pois a maioria dos aparatos tecnológicos anteriores à década de 70 era diretamente controlada por microcontroladores simples e por processos mecânicos). Sejam eles por válvulas, capacitores, motores, entre outros modelos de controle analógicos ou eletromecânicos. Foi a partir do interesse militar que a Tecnologia da informação se consolidou, trazendo novas possibilidades e novos métodos de se programar, principalmente com o advento da internet.

Analisando a linha temporal da Tecnologia da Informação, as linguagens de programação 'C' e 'Assembly' foram as pioneiras quando nos referimos a orientação a objetos e processos eletrônicos. Atualmente elas continuam em uso, considerando que as mesmas foram utilizadas para a construção da arquitetura de praticamente todos os computadores que usamos hoje com sistemas operacionais Windows, Linux ou IOS.

Linguagem C

A linguagem C foi desenvolvida em 1972, por Dennis Ritchie e foi implementada pela primeira vez em um computador que utilizava o sistema operacional UNIX. Ela foi uma evolução da BCPL, de Martin Richards e da Algol 68. Trata-se de uma linguagem compilada e estruturada com elevado nível de portabilidade. Atualmente, a linguagem C pode ser aplicada em diferentes tipos de projetos e é uma das mais populares entre os desenvolvedores.

A popularidade dessa linguagem deve-se, principalmente, ao fato de ser uma linguagem flexível, permitindo que ela seja utilizada no desenvolvimento de diversos tipos de aplicações, desde jogos até poderosos controladores de satélites. Essa versatilidade faz com que os programas em C possam ser executados em diversas plataformas com alterações quase nulas. Ela é eficiente pois possui a capacidade de gerar códigos rápidos, com baixo tempo de execução, economia de memória e de estrutura simples.

O programa em C é constituído por um cabeçalho com as diretivas de compilador, onde colocamos a declaração das variáveis, bibliotecas utilizadas, rotinas etc. Um bloco com as instruções principais e outros blocos de rotina e a documentação do programa.

Exemplo de programa em C:

```c
/*Escreva um programa em C que calcule sua média semestral. */

#include <stdio.h>

int main()
{
    float np1, np2;
    float trab;
    float ms;

    printf("Calculo da media semestral. ");
    printf("\n\n");

    printf("Informe a primeira nota do professor: ");
    scanf("%f", &np1);

    printf("Informe a segunda nota do professor: ");
    scanf("%f", &np2);

    printf("Informe a nota do trabalho: ");
    scanf("%f", &trab);

    ms = (np1 * 4 + trab * 2 + np2 * 4) / 10;

    printf("Media semestral: %2.2f", ms);
    printf("\n\n");

    system("PAUSE");
    return 0;
}
```

Assembly e Assembler

Inicialmente devemos separar o Assembly, que é a linguagem, do Assembler, considerado o "montador", um tipo de compilador para esta linguagem. Criada em meados da década de 50, deu início à segunda geração de linguagens de programação, ainda quando os computadores eram valvulados. Com a evolução tecnológica dos processadores, se fez necessário a criação de um novo modelo mais eficaz de programação a nível de máquina. Foi assim que surgiram as primeiras linguagens chamadas "baixo nível".

A linguagem Assembly se utiliza de códigos mnemônicos (mais fáceis de se decorar) e suas instruções são dadas por comandos que funcionam no formato um-para-um, ou seja, uma instrução do programa equivale a uma operação do computador. Embora isso pareça simplificar a arquitetura da máquina, cada computador possui uma maneira diferente de "montagem", então para cada arquitetura computacional se faria necessário conhecer o código aceito por aquele determinado hardware.

O fato das instruções funcionarem uma a uma, faz com que o fluxo entre a linguagem e a máquina se estruture como uma pilha de comandos formando um registro de segmentos na memória que são executados de maneira ordenada linearmente. Outra forma é utilizar um registro de ponteiro para indicar opções específicas na memória. Todas as operações são conduzidas por registradores de dados que armazenam as instruções de acordo com o tipo solicitado.

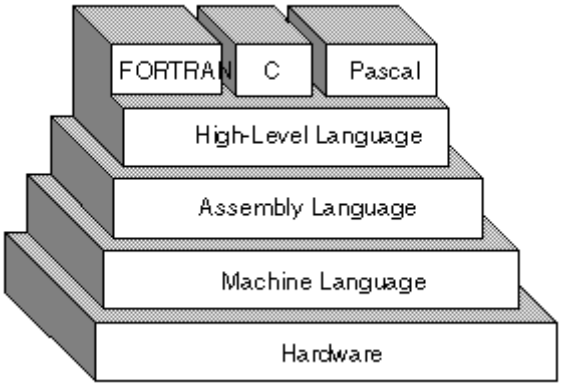
Exemplo de programa básico em Assembly:

```
; Hello World for Intel Assembler (MSDOS)

mov ax,cs
mov ds,ax
mov ah,9
mov dx, offset Hello
int 21h
xor ax,ax
int 21h

Hello:
  db "Hello World!",13,10,"$"
```

A relação entre C e Assembly

Olhando para uma linha do tempo de maneira diacrônica, podemos nos perguntar em qual linguagem estão “escritas” as linguagens computacionais que conhecemos. Ao retornarmos, partindo de C, a resposta seria que a base de C é uma linguagem chamada de B e a cada passo para trás, encontraremos linguagens que já deixaram de ser praticadas, até chegarmos num nível em que não há passo para a ser dado. Quando pensamos em Assembly, considerada uma linguagem de baixo nível, assim chamada por estar sendo executada a um nível mais ‘próximo’ do hardware, estamos praticamente no passo final dessa linha temporal que começamos a trilhar. Isso significa que existe uma co-dependência entre Assembly e C, pois indiretamente uma delas ajudou no embasamento da outra.

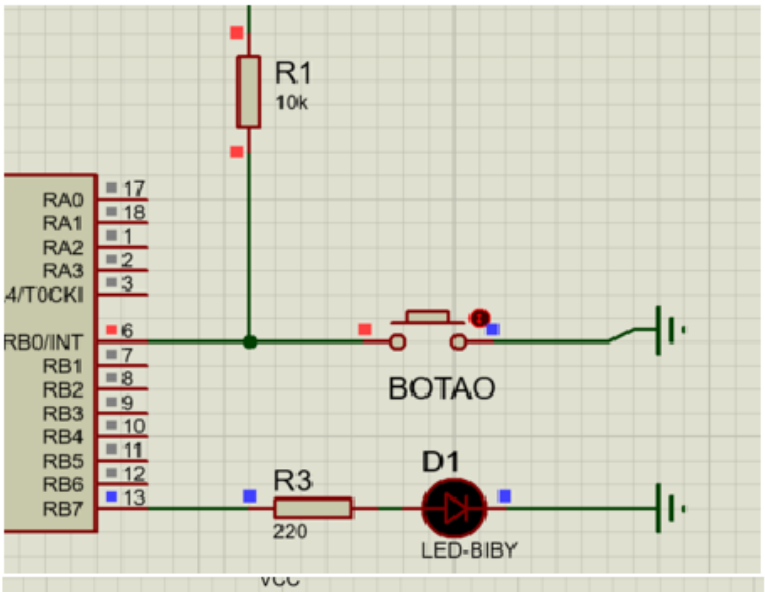
Sendo assim, a primeira relação encontrada é essa herança que a linguagem forneceu de base para a existência da outra. Para além disso, temos uma relação intrínseca nestas linguagens, já que muitos “kernel’s” estão escritos em C. Isso quer dizer que a linguagem de alto nível C é capaz de “chamar” funções do hardware através de Assembly, criando assim muitos programas, aplicações, pacotes e toda a estrutura necessária para que funcione um sistema operacional.



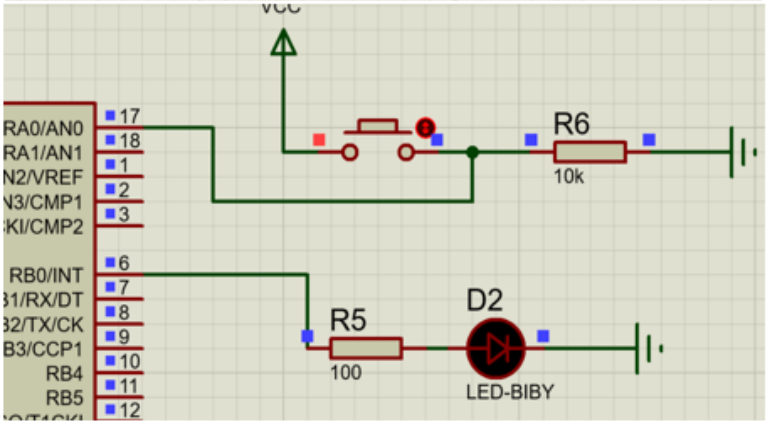
Existem diversas relações possíveis entre estas linguagens: Podemos avaliar uma adequação de quando é mais eficaz determinada linguagem a depender do objetivo final; Há também a possibilidade de analisar as similaridades entre elas, pois é possível desenvolver funções parecidas se utilizando das duas; Existem as relações de dependência, as quais já apresentamos resumidamente. Em específico no experimento, nos foquemos nas dependências de similaridade. A seguir demonstraremos como é possível realizar uma mesma tarefa usando o Assembly e a linguagem C, a fim de observarmos o comportamento dos códigos e seus resultados.

Metodologia

A proposta da aplicação é acendermos um LED por meio de um botão *Pull-up*, em dois microcontroladores de mesma família, usando no primeiro a linguagem Assembly e no exemplo seguinte C. O comparativo se fará com base no comportamento de resposta ao acionarmos o botão, no número de linhas de código, nas funções finais obtidas e no levantamento das análises observadas.



Microcontrolador -Assembly

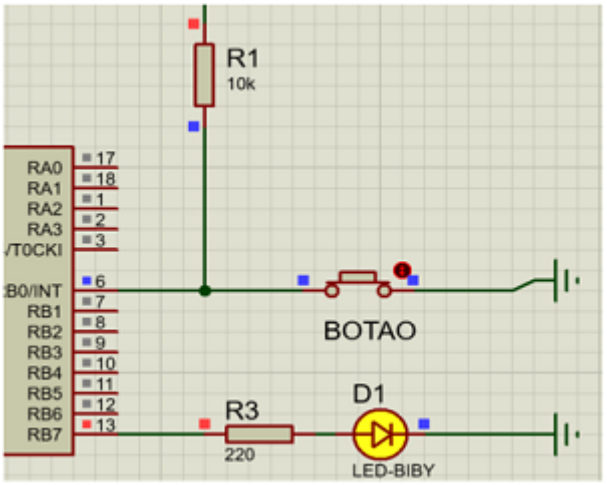


Microcontrolador - C

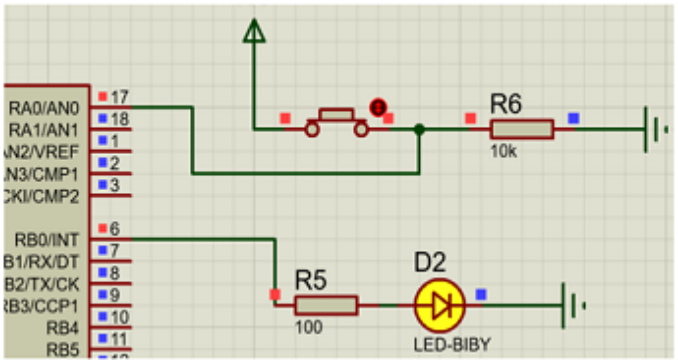
Foram utilizadas no experimento as ferramentas: Proteus 8 Professional; MPlab IDE v8.92; MikroC PRO for PIC. A primeira para os circuitos virtualizados, a segunda para programação em Assembly e a terceira para C. O microcontrolador usado no primeiro exemplo foi um PIC16F84A de 68 Bytes de Ram com clock de 20Mhz e para o exemplo de C foi PIC16F628A de 228 Bytes de Ram e também 20Mhz.

Experimento

Dentro dessa cadeia lógica mostrada nos dois microcontroladores, ao pressionarmos o botão, a resposta seria acender a luz de LED. Para isso se fez necessário a programação dentro dos programas mencionados.



Em assembly, ao pressionar o botão, a luz acende, assim como mostra a figura indicada pelo círculo em 'D1' preenchido da cor amarela. Enquanto o botão não for pressionado, a saída RB0 se mantém positiva e desvia o pulso negativo para a porta RB7. Ao ser pressionado, a corrente positiva em RB0 é desviada para a saída de RB7 acendendo o LED.



Na programação em C existe uma condicional do tipo "IF-ELSE" que identifica e controla qual será o sinal enviado aos polos. Se o botão estiver pressionado ele verifica se RA0 é igual ao nível lógico 1 enviando um sinal positivo a RB0, caso sim. Do contrário ele mantém seu estado.

Resultados

O tempo de resposta das duas execuções foi bem similar, embora o Assembly tenha se mostrado mais eficaz por se comunicar direto com o hardware dos microcontroladores. Não consideramos os tempos de resposta, pois tiveram uma diferença pequena por conta da tarefa simples. Para o exemplo em Assembly, utilizamos 83 linhas de código e a lógica apresentada durante o experimento foi inversamente proporcional, ao que se refere às correntes e botões, comparadas a C.

Em C, a programação foi mais curta e apesar de usarmos 17 linhas, o programa poderia ser feito em menos. A utilização de condicionais na linguagem facilita o processo do programa, mas o microprocessador precisou de ter mais memória RAM para receber instruções nessa linguagem. Ambos os programas rodam em *loop* infinito.

Ambos os experimentos cumpriram com sucesso a função que lhes foi dada. E não observamos nenhuma anormalidade durante o processo.

Conclusão

As duas linguagens observadas oferecem suporte para programarmos a nível de hardware, cada uma com seu benefício e desvantagem. Os estudos demonstram que a curva de aprendizagem em Assembly se mostra um pouco mais lenta para estudantes iniciantes na programação em relação a C.

Das muitas relações existentes entre essas duas linguagens, desde as mais intrínsecas às similaridades comparadas aqui nesse estudo, observamos uma questão fundamental na essência da programação. De que a escolha de determinada linguagem residirá na adequação da função e uso para cada programador e suas necessidades.

Encontramos também nas similaridades a versatilidade de combinar o aprendizado dessas duas linguagens que são exercidas e estudadas ainda nos dias de hoje.

Referências bibliográficas

ANDRADE, Eder. História da computação: Um pouco de Assembly. **Jornal PETnews**. 2012 Disponível em: : <http://www.dsc.ufcg.edu.br/~pet/jornal/maio2014/materias/historia_da_computacao.html> Acessado em: 28 de Junho de 2021.

IC-Unicamp - The Art of Assembly Language (Brief Contents). Documentação Assembly - disponível em: <https://www.ic.unicamp.br/~pannain/mc404/aulas/pdfs/Art%20Of%20Intel%20x86%20Assembly.pdf> Chapter 11 Procedures and Functions, p.565:636. Chapter 15 Strings and Character Sets, p.831:838. Acesso em: 28 de junho de 2021.

Linguagem Assembly. **Wikipédia** a enciclopédia livre. 2020. Disponível em: <https://pt.wikipedia.org/wiki/Linguagem_assembly> Acessado em: 28 de Junho de 2021.

NOLETO, Cairo. Linguagem C: O que é e quais os principais fundamentos! **Blog Trybe**. 2020. Disponível em: <https://blog.betrybe.com/linguagem-de-programacao/linguagem-c/> Acessado em 07 de Julho de 2021.

PEREIRA, Silvio Lago. Linguagem C. São Paulo, USP. 2010.

PINHO, Márcio Sarroglia. Programação C/C++ : História da Linguagem C. Grupo Realidade Virtual, PUCRS. Disponível em: <https://www.inf.pucrs.br/~pinho/LaproI/Historico/Historico.htm> Acessado em: 07 de Julho de 2021.

Projetos com Compilador Micro em C. WebNode ETEC VIRTUAL. São paulo, 2014. Disponível em: <https://etec-virtual7.webnode.com/microcontroladores/projetos-com-compilador-micro-c/> Acessado 10 de Julho de 2021.

RAMBO, Wagner. Botão com LED | Assembly para PIC #006. Youtube, 13 de Dezembro de 2015. Disponível em:

<https://www.youtube.com/watch?v=g8ltH0UUpOQ&ab_channel=WRKits> Acessado em 10 de Julho de 2021.

SCHMIDT, Gerhard. Introdução para o iniciante à linguagem assembly dos microprocessadores ATMEL-AVR. **Avr-Asm-Tutorial**.2003. Disponível em: <https://www.ic.unicamp.br/~ducatte/mc404/2009/docs/beginner_pt.pdf> Acessado em: 28 de Junho de 2021.

STALLINGS, William. Arquitetura e organização de computadores. 10º Edição São Paulo: Pearson Education do Brasil. 2017.