Organização e Arquitetura de Computadores I - Fábio Nakano - 04/04/2023
Atenção!! No dia que estou escrevendo este documento, a prova ainda não foi corrigida. Não me responsabilizo pelos trechos de respostas.

Contexto inicial:

Computadores atuais são suficientemente poderosos para executar compiladores atuais. Compiladores atuais são suficientemente elaborados para partir do código em linguagem C e entregar o executável. Isto permite abstrair (abstract out) muitos conceitos da organização e arquitetura de computadores que deseja-se apresentar, ensinar, aprender e testar (o aprendizado) nesta disciplina. A fim de explicitar esses conceitos, são usadas arquiteturas e ferramentas historicamente relevantes, ainda que, comparativamente, primitivas ou; arquiteturas didáticas. A conexão entre o atual e o primitivo é deixada para o(a) aluno(a).

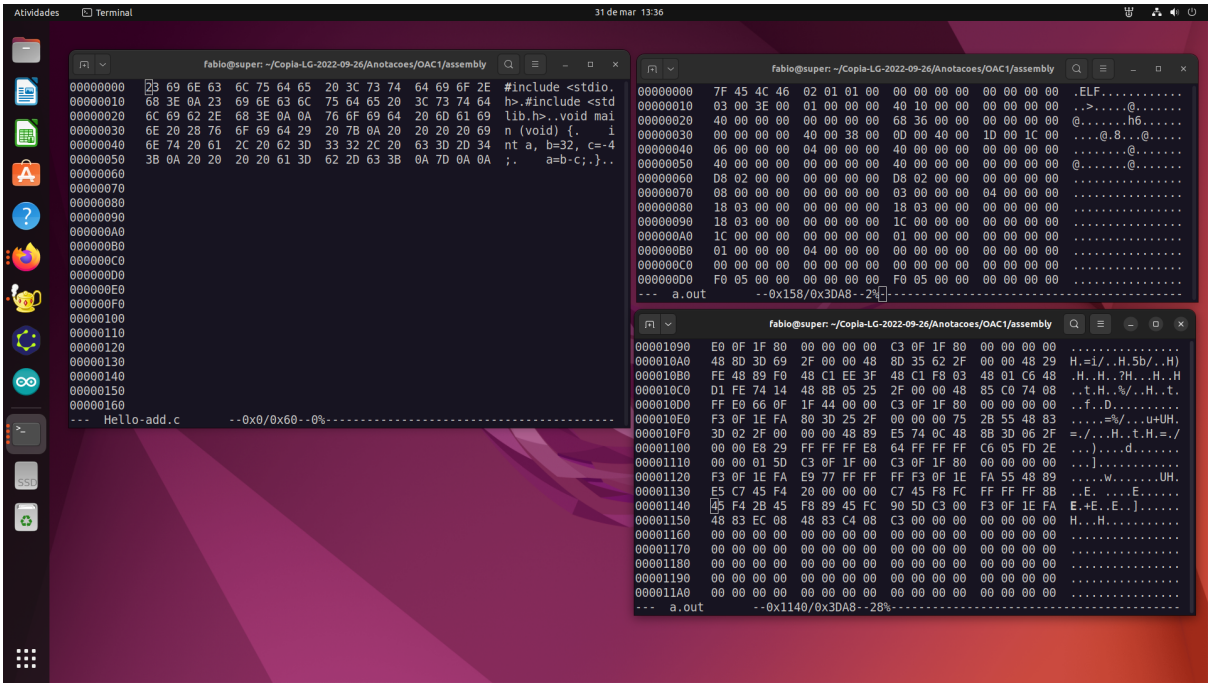
Nesta avaliação serão usados exemplos gerados em programas (compiladores, sistemas operacionais) atuais e os conceitos e exemplos apresentados no livro e no EP para verificar até onde a conexão entre o atual e o primitivo foi alcançada.

Contexto a adicionar para as questões que se seguem:

No ciclo de vida (Dev Ops), considere os passos: (1) Codificar, (2) Compilar, (3) Executar; de um ponto de vista prático (ié, sobre um exemplo).
O programa do exemplo pode "simplesmente" escrever Olá na tela. Para ser mais simples ainda, o programa "apenas" cria três variáveis, inicializa duas e armazena a subtração dessas duas na terceira. A linguagem do código-fonte é C.

O programa é compilado com a linha de comando "gcc Ola-add.c", o resultado é "a.out". "a.out" pode ser executado com a linha de comando "./a.out". Ola-add.c não pode ser imediatamente executado.

Podemos editar Ola-add.c e a.out usando um editor hexadecimal, por exemplo, "hexedit". A captura de tela /home/fabio/Documentos/OAC1-2023/ProvaOnline1/Captura-de-tela-1.png mostra trechos dos arquivos. À esquerda hexedit editando "Ola-add.c" e à direita editando "a.out", acima o começo de "a.out", abaixo um outro trecho de "a.out" tomado propositalmente. Isto pode ser repetido com os arquivos anexados na avaliação



Para uniformizar a representação pela mais primitiva, convencionemos que o conteúdo dos arquivos são os números (hexadecimais) do lado esquerdo da tela do hexedit.

Questão 1: Quais são as semelhanças e as diferenças entre "Ola-add.c" e "a.out"? Sabe-se que há algumas equivalências entre elementos de "Ola-add.c" e "a.out" pois são diferentes representações do mesmo algoritmo. Apenas com a informação da captura de tela é possível mostrar algum elemento correspondente? No contexto do processador, como é possível saber se um arquivo pode ser executado? No contexto do computador, o que nos impede de executar "Ola-add.c"? No contexto do processador, "Ola-add.c" poderia ser executado? Elabore sobre isso em sua resposta.

Contexto adicional:

O "gcc" pode ser usado para gerar código em linguagem intermediária. No caso, a linguagem de montagem da particular arquitetura em que foi instalado. Outra forma de obter o código em linguagem intermediária é "desmontando" (diassembly) o executável, por exemplo, com o programa "objdump". O comando "gcc -S Ola-add.c" gerou o arquivo "Ola-add.s", o comando "objdump -d a.out > Ola-add.dump" gerou o arquivo "Ola-add.dump".

Resposta:

"Ola-add.c" é um arquivo de código-fonte escrito em linguagem de programação C, enquanto "a.out" é um arquivo executável gerado a partir do código-fonte compilado. A principal diferença entre eles é que o primeiro é um arquivo de texto legível por humanos, que contém as instruções em linguagem de programação C, enquanto o

segundo é um arquivo binário que contém instruções em linguagem de máquina que podem ser executadas diretamente pelo processador.

Algumas equivalências entre "Ola-add.c" e "a.out" podem ser identificadas a partir da captura de tela, como o fato de ambos conterem a função "main", que é a função principal do programa e é executada quando o programa é iniciado, mas é bastante difícil apontar semelhanças entre ambos apenas pela captura de tela, tendo em vista a grande diferença de linguagens de ambas.

No contexto do processador, é possível saber se um arquivo pode ser executado verificando se ele contém instruções em linguagem de máquina que são compatíveis com a arquitetura do processador. Se não houverem instruções compatíveis, o programa não poderá ser executado.

Já no contexto do computador, o que nos impede de executar "Ola-add.c" é o fato de que ele não contém instruções em linguagem de máquina. Para executar o programa, é necessário compilar o código-fonte em linguagem de programação C para gerar um arquivo executável contendo instruções em linguagem de máquina que possam ser executadas pelo processador.

Por fim, no contexto do processador, "Ola-add.c" não pode ser executado diretamente, pois ele contém instruções em linguagem de programação C, que precisam ser compiladas para gerar um arquivo executável contendo instruções em linguagem de máquina que possam ser executadas pelo processador.

Questão 2: Que partes desses arquivos correspondem à função "main" de "Ola-add.c"? (copie e cole as linhas dos respectivos arquivos, indicando a que arquivo pertencem e explique como você chegou a essa correspondência).

Contexto adicional:

A saída de "objdump" pode ser vista como formatada em colunas. A primeira coluna corresponde ao índice ou endereço ou posição do primeiro byte da segunda coluna. A segunda coluna lista bytes presentes em "a.out". Da terceira coluna em diante contém a "desmontagem" dos bytes, apresentando os comandos em linguagem de montagem.

Resposta:

Para identificar a parte do código correspondente à função "main" de "Ola-add.c", podemos usar a "desmontagem" gerada pelo comando "objdump -d a.out". A função "main" é a primeira a ser executada quando um programa é iniciado e, portanto, é geralmente o ponto de entrada para o programa. Na "desmontagem", a função "main" é identificada pelo rótulo "_start" que é o ponto de entrada do programa no nível de montagem.

Podemos encontrar a correspondência da seguinte forma:

No arquivo "Ola-add.c", a função "main" é definida a partir da linha 3 até a linha 7.

No arquivo "Ola-add.s", podemos ver a representação em linguagem de montagem do código C gerado pelo compilador. A função "main" começa na linha 8 com o rótulo "_main:". Essa é a representação em linguagem de montagem do código C da função "main".

No arquivo "Ola-add.dump", podemos ver a "desmontagem" do arquivo "a.out" em linguagem de montagem. Podemos identificar a correspondência da função "main" a partir da linha que contém o rótulo "_start". A partir desse ponto, o código de montagem chama a função "_main", que é a função "main" do programa em linguagem C.

Portanto, a correspondência da parte do código que representa a função "main" é:

Arquivo "Ola-add.c": linhas 3 a 7.

Arquivo "Ola-add.s": linha 8.

Arquivo "Ola-add.dump": a partir da linha com o rótulo "_start" e a linha que contém a chamada para a função "_main".

Questão 3: Que correspondências há entre a saída de "objdump" (exemplo em arquitetura real e atual), o enunciado do EP1 (exemplo em arquitetura didática - Simulador do HIPO, seções: codificação de instruções, instruções do computador HIPO, Problema 1, Representações internas) e o IAS (exemplo em arquitetura historicamente relevante - Livro texto, figuras 1.6, 1.7, Tabela 1.1)

Contexto adicional:
HIPO é um "computador hipotético". Criado para fins didáticos.

Construir um simulador do HIPO implica no contato com instruções em linguagem de máquina, sua correspondência com instruções em linguagem de montagem, como instruções e dados são armazenados na memória.

Algumas estruturas internas, como os registradores são apresentados.

O simulador, tomado como um modelo do HIPO, torna-se um ponto de partida para outros modelos, por exemplo, mais próximos de uma implementação física, elicitando, progressivamente, os elementos da organização do computador e os sinais elétricos que transitam entre os elementos.

Também é possível criar um montador: um programa que receba um programa codificado na linguagem de montagem do HIPO e que gere uma imagem da memória que, quando carregada no HIPO, o faz "executar o programa codificado na linguagem de montagem. https://www.ime.usp.br/~jstern/miscellanea/MaterialDidatico/hipo.htm

Resposta:
A saída do comando "objdump" mostra a representação em linguagem de montagem de um arquivo executável. Já o enunciado do EP1 apresenta o simulador do HIPO, um computador hipotético usado para fins educacionais, que descreve a linguagem de montagem utilizada e as instruções disponíveis, além das representações internas dos dados. O IAS é uma arquitetura histórica relevante, descrita em um livro-texto, que apresenta figuras e tabelas para explicar o funcionamento da máquina, suas instruções, registradores e formas de armazenamento de dados.

Embora essas fontes de informação apresentem abordagens diferentes para descrever as arquiteturas de computadores, é possível encontrar algumas correspondências entre elas. Por exemplo, o IAS e o simulador do HIPO possuem registradores e instruções semelhantes em sua arquitetura, enquanto a saída do "objdump" descreve a desmontagem das instruções em linguagem de montagem de um arquivo executável real.

Em resumo, cada uma dessas fontes fornece uma abordagem diferente para entender a organização e arquitetura de computadores, mas é possível encontrar correspondências entre elas que ajudam a formar uma visão mais ampla e integrada sobre o assunto.