1 – MÁQUINAS VIRTUAIS, MÁQUINAS MULTINÍVEL E LINGUAGENS

1.1 - INTRODUÇÃO

Um computador digital é uma máquina capaz de nos solucionar problemas através da execução de instruções que lhe são fornecidas. Denomina-se programa uma seqüência de instruções que descreve como executar uma determinada tarefa. Os circuitos eletrônicos de cada computador podem reconhecer e executar diretamente um conjunto limitado de instruções simples para as quais todos os programas devem ser convertidos antes que eles possam ser executados. Estas instruções básicas raramente são mais complicadas do que:

Some dois números. Verifique se um número é zero.

Mova um dado de uma parte da memória do computador para outra.

Juntas, as instruções primitivas do computador formam uma linguagem que torna possível as pessoas se comunicarem com o computador. Tal linguagem é denominada linguagem de máquina. Ao se projetar um novo computador deve-se decidir que instruções devem estar incluídas nesta linguagem de máquina. Geralmente, tenta-se fazer as instruções primitivas tão simples quanto possível, consistentes com o uso pretendido e necessidades de desempenho, a fim de reduzir a complexidade e o custo da eletrônica empregada. Como a maioria das linguagens de máquina são muito simples, é difícil e tedioso utilizá-las.

Este problema pode ser atacado de duas maneiras principais, ambas envolvendo o projeto de um novo conjunto de instruções de uso mais conveniente para as pessoas do que o conjunto de instruções embutidas da máquina. Juntas, estas novas instruções formam uma linguagem que chamaremos de L2, exatamente como as instruções embutidas de máquina formam uma linguagem, que chamaremos de L1. Os dois conceitos diferem na maneira com que os programas escritos em L2 são executados pelo computador, que, afinal de contas, pode apenas executar programas escritos em sua linguagem de máquina, L1.

Um método de execução de um programa escrito em L2 consiste, em primeiro lugar, em substituir cada instrução nele por uma seqüência equivalente de instruções em L1. O programa resultante, composto inteiramente de instruções L1. O computador então executa o novo programa em L1 em vez do programa em L2. Denomina-se esta técnica tradução.

A outra técnica consiste em escrever um programa em L1 que receba os programas escritos em L2 como dados de entrada e efetue a execução examinando uma instrução de cada vez e executando a seqüência equivalente de instruções L1 diretamente. Esta técnica não requer a geração de um novo programa em L1. Denomina-se interpretação, e o programa que a executa denomina-se interpretador.

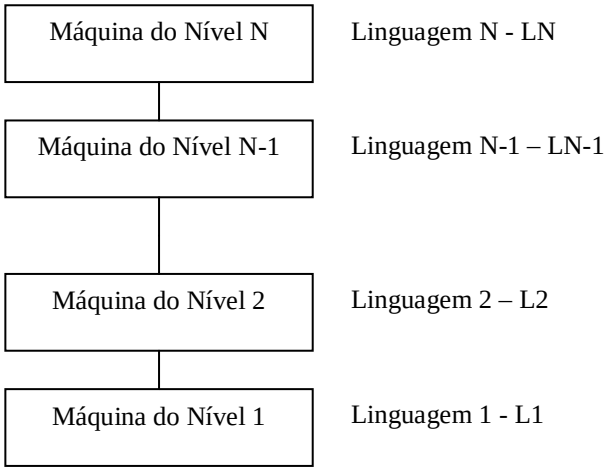
A tradução e a interpretação são similares. Em ambos os métodos, as instruções em L2 são executadas pelas seqüências equivalentes de instruções em L1. A diferença é que, na tradução, o programa inteiro em L2 é primeiramente convertido para um programa em L1, o programa L2 é abandonado, e o novo programa em L1 é executado. Na interpretação, depois de cada instrução L2 ser examinada e decodificada, ela é executada imediatamente. Nenhum programa traduzido é gerado. Ambos os métodos são amplamente utilizados.

Em vez de pensar em termos de tradução e interpretação, é muitas vezes mais adequado imaginar a existência de um computador hipotético ou máquina virtual cuja linguagem de máquina seja L2. Se tal máquina pudesse ser construída a um custo suficientemente baixo, não haveria absolutamente nenhuma necessidade de se ter L1 ou uma máquina que executasse programas em L1. As pessoas poderiam simplesmente escrever seus programas em L2 e o computador os executaria diretamente. Até mesmo se a máquina virtual cuja linguagem é L2 fosse muito cara para ser construída com circuitos eletrônicos, poder-se-ia ainda escrever programas para ela. Estes programas podem ser interpretados ou traduzidos por um programa escrito em L1, linguagem esta que possa ser diretamente executada por um computador existente. Em outras palavras, podem-se escrever programas para máquinas virtuais como se elas realmente existissem.

Para tornar práticas a tradução e a interpretação, as linguagens L1 e L2 não devem ser "muito" diferentes. Esta restrição significa muitas vezes que L2, embora melhor que L1, estará ainda longe do ideal para a maioria das aplicações. Talvez isso seja desencorajante em vista do objetivo original de criar L2, ou seja, livrar o programador da carga de ter que expressar os algoritmos em uma linguagem mais adequada às máquinas do que às pessoas. Entretanto, a situação está longe de desesperante.

A solução óbvia é criar um outro conjunto de instruções que seja mais dirigido às pessoas e menos dirigido à máquina do que aquele de L2. Este terceiro conjunto também constitui uma Linguagem, que chamaremos L3. Podem-se escrever programas em L3 como se uma máquina virtual cuja Linguagem de máquina fosse L3 realmente existisse. Tais programas podem tanto ser traduzidos para L2 quanto executados por um interpretador escritos em L2.

A invenção de toda uma série de linguagens, cada uma delas mais conveniente do que suas predecessoras, pode continuar indefinidamente até que uma adequada seja finalmente conseguida. Cada Linguagem utiliza sua predecessora como base, assim podemos visualizar um computador que utiliza esta técnica como uma série de camadas ou níveis, um sobre o outro, como está mostrado na figura abaixo. A linguagem ou nível mais baixo é a(o) mais simples e a Linguagem ou nível mais alto é a(o) mais sofisticada(o).



1.2 - LINGUAGENS, NÍVEIS E MÁQUINAS VIRTUAIS

Há uma relação importante entre uma linguagem e uma máquina virtual. Cada máquina tem a sua linguagem de máquina, que consiste em todas as instruções que a máquina pode executar. De fato, uma máquina define uma linguagem. Do mesmo modo, uma linguagem define uma máquina - ou seja, a máquina que pode executar todos os programas escritos nesta linguagem. Naturalmente, a máquina definida por uma determinada linguagem pode ser enormemente complicada e cara de construir diretamente a partir de circuitos eletrônicos, mas de qualquer modo podemos imaginá-la. Uma máquina com C, Pascal ou Cobol como sua linguagem de máquina seria complicadíssima, mas, certamente imaginável e, talvez dentro de alguns anos, tal máquina seja considerada de construção trivial.

Um computador com ***n*** níveis pode ser considerado como ***n*** diferentes máquinas virtuais, cada uma delas com uma diferente linguagem de máquina. Podemos utilizar os termos "nível" e "máquina virtual" indistintamente. Apenas os programas escritos em linguagem L1 podem ser executados diretamente pelos circuitos eletrônicos, sem a necessidade de intervenção da tradução ou interpretação. Os programas escritos em L2, L3, ... Ln, devem ser interpretados por um interpretador em execução em um nível inferior ou ser traduzidos para uma outra linguagem correspondente no nível inferior.

Uma pessoa cujo trabalho fosse escrever programas para a máquina virtual de nível **n** não precisaria estar ciente dos interpretadores e tradutores subjacentes. A estrutura da máquina assegura que estes programas serão de algum modo executados. É de pouco interesse se eles são executados passo a passo por um interpretador, que por sua vez é executado por um outro interpretador, ou se eles são executados diretamente pelos circuitos eletrônicos. O mesmo resultado surge em ambos os casos: os programas são executados.

A maioria dos programadores que utilizam uma máquina de nível **n** está apenas interessada no nível do topo, que lembra o mínimo possível a linguagem de máquina situada lá embaixo. Entretanto, os interessados em compreender como um computador realmente funciona devem estudar todos os níveis. As pessoas interessadas em projetar novos computadores ou novos níveis (isto é, novas máquinas virtuais) devem também estar familiarizadas com outros níveis além do nível do topo. Os conceitos e técnicas de construção de máquinas como uma série de níveis e os detalhes de alguns níveis importantes constituem por si são o assunto principal deste texto.

Organização Estrutural de Computadores vem do fato de se visualizar o computador como um conjunto hierárquico de níveis que provê uma boa estrutura para compreender como os computadores estão organizados. Além disso, projetar computadores como uma série de níveis ajuda a assegurar um produto final bem estruturado.

1.3 MÁQUINAS MULTINÍVEL CONTEMPORÂNEAS

A maioria dos computadores modernos possui dois ou mais níveis. As máquinas de seis níveis são muito comuns, como está mostrado na figura a seguir. O nível 0, lá embaixo, é o hardware verdadeiro da máquina. Seus circuitos executam os programas em linguagem de máquina de nível 1. Para completar, deve-se mencionar ainda a existência de um outro nível abaixo do nível 0. Este nível, não está mostrado na figura porque se situa no campo da engenharia elétrica (e por isto está fora do escopo deste texto), é denominado nível de dispositivos. A este nível, o projetista enxerga transistores individuais, os quais são os primitivos de mais baixo nível para os projetistas de computadores. (Naturalmente, pode-se também indagar como funcionam os transistores por dentro, mas isto já pertence à física de estado sólido).

No nível mais baixo que estudaremos, o nível da lógica digital, os objetos interessantes são denominados portas. Embora construídas a partir de componentes analógicos, como os transistores, as portas podem ser modeladas com precisão como dispositivos digitais. Cada porta possui uma ou mais entradas digitais (sinais representando 0 ou 1), e fornece como saída funções simples destas entradas, tais como AND ou OR. Cada porta é constituída no máximo por um punhado de transistores. Embora o conhecimento do nível de dispositivos seja um tanto especializado, com o advento dos microprocessadores e microcomputadores, mais e mais pessoas estão entrando em contato com o nível de lógica digital.

O nível superior seguinte, o nível 1, que é o verdadeiro nível de linguagem de máquina. Ao contrário do nível 0, onde não existe o conceito de programa como uma seqüência de instruções a serem executadas, em nível 1 há definitivamente um programa, denominado microprograma, cuja função é interpretar as instruções de nível 2. Chamamos o nível 1 de nível de microprogramação. Embora seja verdadeiro que não existem dois computadores com níveis de microprogramação idênticos, existem semelhanças suficientes que nos permitem extrair os aspectos essenciais do nível e discutí-Io como se ele fosse bem definido. Por exemplo, poucas são as máquinas que têm mais de 30 instruções este nível, e a maior parte destas instruções envolve a movimentação de dados de uma parte da máquina para outra, ou a realização de alguns testes simples. Cada máquina de nível 1 tem um ou mais microprogramas que são executados nela. Cada microprograma define implicitamente uma linguagem de nível 2 (e uma máquina virtual, cuja linguagem de máquina, essa linguagem). Estas máquinas de nível 2 têm muito em comum. Até mesmo máquinas de nível 2 de fabricantes diferentes têm mais semelhanças do que diferenças. Neste texto, denominaremos este nível de nível de máquina convencional, por falta de um nome de aceitação geral.

Todo fabricante de computadores pública um manual para cada tipo de computador que ele comercializa, intitulado "Manual de referência da Linguagem de Máquina" ou "Princípios de Operação do Computador Western Combat Modelo 100X" ou algo semelhante. Estes manuais tratam, na realidade, da máquina virtual de nível 2, e não da máquina real de nível 1. Quando eles descrevem o conjunto de instruções de máquina, estão, de fato, descrevendo as instruções interpretadas pelo microprograma, e não as próprias instruções de hardware. Se o fabricante de computadores escrever para uma de suas máquinas dois interpretadores que interpretem duas máquinas diferentes de nível 2, ele precisará fornecer dois manuais de referência de "Linguagem de máquina", um para cada interpretador.

Deve-se mencionar que alguns computadores não têm um nível de microprogramação. Nestas máquinas, as instruções do nível de máquina convencional são executadas diretamente pelos circuitos eletrônicos (nível 0), sem qualquer interpretador intermediário de nível 1. Como resultado, o nível 1, e não o nível 2, é o nível de máquina convencional. De qualquer modo, continuaremos a chamar o nível de máquina convencional de "nível 2", apesar destas exceções.

O terceiro nível , geralmente um nível híbrido. A maior parte das instruções desta linguagem está também na linguagem de nível 2. (Não há motivo para uma instrução que aparece em um nível não poder estar presente também em outros níveis.) Além disso, há um conjunto de novas instruções, uma organização diferente de memória, capacidade de execução de dois ou mais programas em paralelo, e várias outras características. Existem diferenças maiores entre máquinas de nível 3 do que entre máquinas de nível 1 ou de nível 2.

As novas facilidades adicionadas ao nível 3 são efetuadas por um interpretador em execução no nível 2, que historicamente tem sido denominado sistema operacional. Aquelas instruções de nível 3 idênticas às do nível 2 são executadas diretamente pelo microprograma, e não pelo sistema operacional. Em outras palavras, algumas das instruções de nível 3 são interpretadas pelo sistema operacional e outras são interpretadas diretamente pelo microprograma. Isto é, o que se quer dizer por "híbrido". Denominaremos este nível de nível de sistema operacional.

Há uma diferença fundamental entre os níveis 3 e 4. Os três níveis inferiores não foram projetados para o uso direto pelo programador médio comum. Eles são voltados principalmente para a execução dos interpretadores e tradutores necessários para suportar os níveis superiores. Estes interpretadores e tradutores são escritos por pessoas chamadas de programadores de sistema, que são especialistas em projetar e implementar novas máquinas virtuais. Os níveis 4 e superiores são dirigidos aos programadores de aplicação com problemas a serem solucionados.

Uma outra mudança que ocorre no nível 4 é o método pelo qual os níveis superiores são suportados. Os níveis 2 e 3 são sempre interpretados. Os níveis 4, 5 e superiores são geralmente, mas nem sempre, suportados por tradução.

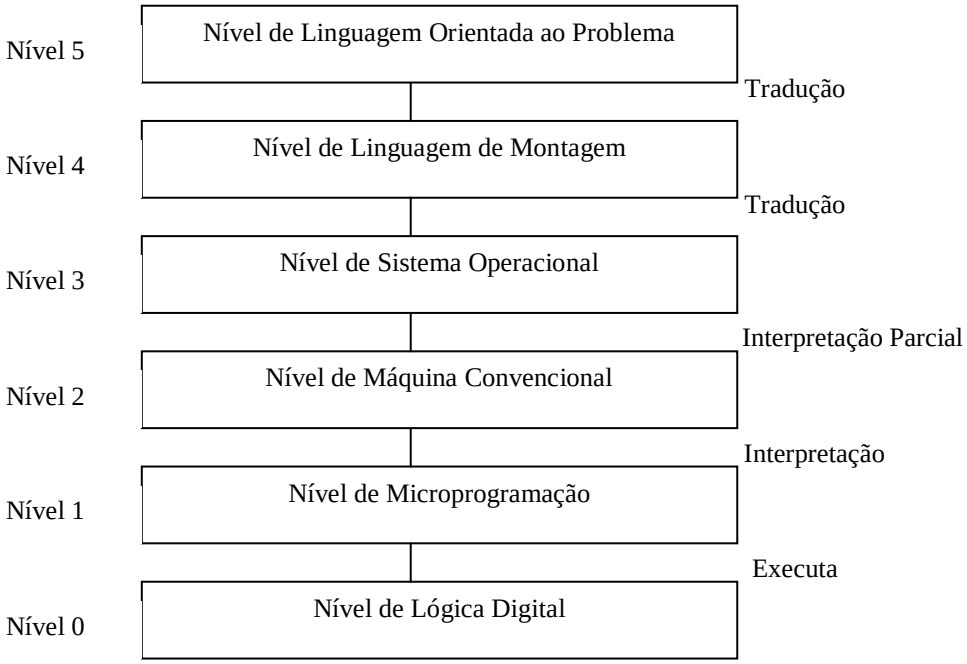
Uma outra diferença ainda entre os níveis 1, 2 e 3, de um lado, e os níveis 4, 5 e superiores, de outro, é a natureza da linguagem provida. As linguagens de máquina de níveis 1, 2 e 3 são numéricas. Os programas escritos nestas linguagens consistem em longas séries de números, os quais são ótimos para as máquinas e péssimos para as pessoas. A partir do nível 4, as linguagens contém palavras e abreviaturas significativas para as pessoas.

O nível 4, o nível da linguagem de montagem, é realmente uma forma simbólica para uma Linguagem subjacente. Este nível provê um método para as pessoas escreverem programas para os níveis 1, 2 e 3 de uma maneira não tão desagradável quanto as próprias linguagens de máquina virtual. Os programas em Linguagem de montagem são primeiramente traduzidos para a Linguagem do nível 1, 2 ou 3, e então são interpretados pela máquina virtual ou real apropriada. O programa que executa a tradução é denominado montador.

O nível 5 consiste em linguagens projetadas para serem utilizadas por programadores de aplicação com problemas a serem resolvidos. Tais linguagens são denominadas linguagens de alto nível. Existem literalmente centenas delas. Algumas das mais conhecidas são C, C++, Java, Pascal,

Delphi. Os programas escritos nestas linguagens são geralmente traduzidos para o nível 3 ou nível 4 por tradutores conhecidos como compiladores, embora ocasionalmente eles sejam interpretados.

Os níveis 6 e superiores consistem em coleções de programas projetados para criar máquinas especialmente adequadas para certas aplicações. Eles contêm grandes quantidades de informação acerca da aplicação. Podem-se imaginar máquinas virtuais voltadas a aplicações em administração, educação, projeto de computadores etc.



Em resumo, o ponto-chave a ser lembrado é que os computadores são projetados como uma série de níveis, cada um deles construído sobre seu predecessor. Cada nível representa uma abstração distinta, com a presença de diferentes objetos e operações. Ao projetar e analisar computadores deste modo, podemos temporariamente suprimir detalhes irrelevantes e assim reduzir um assunto complexo para algo mais fácil de ser compreendido.

O conjunto de tipos de dados, operações e características de cada nível é denominado sua arquitetura. A arquitetura trata daqueles aspectos que são visíveis ao usuário daquele nível. Os aspectos que um programador vê, tais como quanta memória está disponível, fazem parte da arquitetura. Aspectos de implementação, tais como qual o tipo de tecnologia da pastilha utilizada na implementação da memória, não fazem parte da arquitetura. O estudo de como projetar as partes de um computador que são visíveis aos programadores denomina-se arquitetura de computadores. Na prática, arquitetura de computadores e organização de computadores significam essencialmente a mesma coisa.

1.4 HARDWARE, SOFTWARE E MÁQUINAS MULTINÍVEL

Os programas escritos na linguagem de máquina de um computador (nível 1) podem ser executados diretamente pelos circuitos eletrônicos do computador (nível 0), sem quaisquer interpretadores ou tradutores intermediários. Estes circuitos eletrônicos, juntamente com a memória e dispositivos de entrada/saída, constituem o hardware do computador. O hardware é composto de objetos tangíveis - circuitos integrados, placas de circuito impresso, cabos, fontes de alimentação, memórias, leitoras de cartões, impressoras e terminais - em lugar de idéias abstratas, algoritmos ou instruções.

O software, ao contrário, consiste em algoritmos (instruções detalhadas que dizem como fazer algo) e suas representações para o computador - ou seja, os programas. Os programas podem estar

representados em cartões perfurados, fita magnética, filme fotográfico e outros meios, mas a essência do software está no conjunto de instruções que constitui os programas, não nos meios físicos sobre os quais eles estão gravados.

Uma forma intermediária entre o hardware e o software é o firmware, que consiste no soflware embutido em dispositivos eletrônicos durante a fabricação. O firmware é utilizado quando se espera que os programas raramente ou nunca serão mudados, por exemplo, em brinquedos ou instrumentos. O firmware é também usado quando os programas não podem ser perdidos ao se acabar a alimentação (p. ex., quando a bateria da boneca se descarrega). Em muitos computadores o microprograma está em firmware.

Um tema central que sempre aparecerá é:

Hardware e software são logicamente equivalentes.

Qualquer operação efetuada pelo software pode também ser implementada diretamente em hardware, e qualquer instrução executada pelo hardware pode também ser simulada pelo software.

A decisão de se colocar certas funções em hardware e outras em software baseia-se em fatores tais como: custo, velocidade, confiabilidade e freqüência esperada de alterações. Não há regras rígidas e diretas para se dizer que X deve ser implementado em hardware e Y deve ser programado explicitamente. Projetistas com objetivos diferentes podem, e muitas vezes o fazem, tomar diferentes decisões.

Nos primeiros computadores, a distinção entre hardware e software era clara. O hardware executava algumas instruções simples, tais como ADD (somar) e JUMP (saltar para), e tudo o mais era programado explicitamente. Se um programa precisasse multiplicar dois números, o programador tinha que escrever seu próprio procedimento de multiplicação ou pedir um emprestado à biblioteca. Com o passar do tempo, tornou-se claro aos projetistas de hardware que certas operações eram executadas com uma freqüência suficiente para justificar a construção de circuitos especiais para executá-las diretamente em hardware (para torná-las mais rápidas). Isto resultou em uma tendência de se levar as operações para os níveis inferiores. O que era previamente programado explicitamente no nível de máquina convencional era mais tarde encontrado abaixo dele, no hardware.

Com a chegada da microprogramação e de computadores multinível, a tendência oposta também ocorreu. Nos primeiros computadores, não havia dúvidas de que a instrução ADD era executada diretamente pelo hardware. Em um computador microprogramado, a instrução ADD do nível de máquina convencional era interpretada através de um microprograma sendo executado no nível situado mais ao fundo, e era executada como uma série de pequenos passos: buscar a instrução, descobrir seu tipo, localizar os dados a serem adicionados, buscar os dados da memória, executar a adição e armazenar o resultado. Este é um exemplo de função que moveu para cima, do nível de hardware para o de microprograma. Nós enfatizaremos mais uma vez que: não existem regras rígidas e diretas a respeito do que deve estar em hardware e do que deve estar em software.

Ao desenvolverem uma máquina multinível, os projetistas devem decidir o que colocar em cada nível. Esta é uma generalização do problema mencionado anteriormente, de se decidir o que colocar em hardware e o que colocar em software, o hardware sendo meramente de nível mais baixo. É interessante notar que algumas das características de certos computadores modernos são agora executadas pelo hardware ou pelo microprograma, mas que originalmente foram programadas explicitamente no nível de máquina convencional. Dentre elas estão incluídas:

1. Instruções para multiplicação e divisão de inteiros.
2. Instruções aritméticas de ponto-flutuante.
3. Instruções aritméticas de dupla precisão (aritmética de números com duas vezes o número usual de algarismos significativos).
4. Instruções para chamar e retomar de procedimentos.

5. Instruções para acelerar laços (looping). '
6. Instruções de contagem (somar 1 a uma variável).
7. Instruções para manipular cadeias de caracteres.
8. Aspectos de aceleração de computações envolvendo matrizes (indexação e endereçamento indireto).
9. Características para permitir que os programas fossem movidos na memória após terem começado a ser executados (facilidades de recolocação).
10. Relógios para temporizar programas.
11. Sistemas de interrupção que avisem o computador tão logo uma operação de entrada ou saída esteja completada.
12. A capacidade de parar um programa e iniciar outro utilizando poucas instruções (chaveamento de processos).

O objetivo da discussão é mostrar que a fronteira entre o hardware e o software é arbitrária e que varia constantemente. O software de hoje é o hardware de amanhã, e vice-versa. Além disso, os limites entre os vários níveis são fluidos. Do ponto de vista do programador, a maneira como uma instrução está realmente implementada não é importante (exceto, talvez, com relação à velocidade). Alguém que estiver programado no nível de máquina convencional pode usar sua instrução de multiplicação como se esta fosse uma instrução de hardware sem se preocupar com isto, ou até mesmo estar ciente se ela é realmente uma instrução de hardware ou não.

O fato de o programador não estar ciente de como o nível que ele está utilizando é implementado leva ao conceito de projeto estruturado de máquinas. Um nível é muitas vezes chamado de máquina virtual porque o programador pensa nele como uma máquina física real, mesmo se ela não existir realmente. Estruturando uma máquina como uma série de níveis, os programadores que trabalham no nível **n** não precisam saber de todos os detalhes confusos dos níveis inferiores. Esta estruturação simplifica enormemente a produção de máquinas (virtuais) complexas.

Fonte:
TANENBAUM,A. Organização Estruturada de Computadores. 4ª edicão. Rio de janeiro: LTC, 2006.