**CAPÍTULO 1 PATTERSON**

* 3ª revolução provocada pelos PCs.
* Várias aplicações tornaram-se viáveis graças aos avanços tecnológicos: caixas automáticos, computação em automóveis, laptops, projeto genoma humano, WWW.
  1. – Classes de aplicações de computador e suas características.

Desktops

* Mais conhecida classe
* Uso pessoal
* Enfatizam o bom desempenho a um único usuário
* Baixo custo
* Usados para software shrik-wrap, independente.
* Incorpora alguns dispositivos de I/O.

Servidores

* Mais caros.
* Versçao moderna dos mainframes.
* Projetados para suportar grande carga de trabalho, seja realizando uma atividade complexa ou muitas atividades simples ao mesmo tempo.
* Mesma tecnologia básica dos desktops, mas com possibilidade de extensão de processamento e entrada e saída.
* Dão ênfase a estabilidade.
* Menor fração.
* Baseadas em software de outra origem.
* Acesso geralmente pela rede.

Computadores embutidos

* Maior classe
* Faixa mais ampla de aplicações e desempenho.
* A aplicação tem limitações quanto ao custo e a potência.
* As aplicações são projetadas para cumprir necessidades especificas.
* Executam uma ação relacionada a um único sistema.
* O usuário mal os vê.
* Menor tolerância a falha.
* Baixo custo.
* Simplicidade ajuda a melhorar a falha.
* Redundante as vezes.

OBS>> Muitos processadores embutidos usam núcleos de processadores que são uma versão de um processador escrito em HDL. Permite a integração do núcleo com o hardware especifico para uma aplicação.

1.2 – Por baixo do seu programa.

* O hardware só pode executar instruções de baixo nível.
* Existem várias camadas entre as aplicações complexas e o baixo nível.
* Essa interface é feita através de vários softwares de sistema, cujos mais importantes são o compilador e o SO.
* Sistema operacional: Faz a interface entre o programa de usuário e o hardware e disponibiliza diversos serviços de supervisão. Por exemplo, manipula entrada e saída, controla o armazenamento e gerenciamento de memória, e faz o compartilhamento do processamento com as diversas aplicações em execução.
* Compiladores: Ele traduz um programa em alto nível para instruções que o hardware possa executar.

De uma linguagem de alto nível para a linguagem do hardware.

* A linguagem de máquina é baseda em dois símbolos, 0 e 1.
* As instruções são apenas grupos de bits.
* Um montador traduz a versão simbólica de uma instrução para números binários.
* Linguagem assembly é a linguagem simbólica para representar instruções binárias. Uma linha por instrução, obrigando o programador a pensar como uma máquina.
* Linguagens de alto nível são compostas por palavras e notações algébricas que podem ser traduzidas por um compilador para a linguagem assembly. Permite que o programador pense em uma linguagem mais natural. A linguagem pode ser projetada de acordo com o uso pretendido. Maior produtividade do programador, concisão. Os programas são independentes do computador onde estão sendo desenvolvidos.

1.3 – Sob as tampas

* O hardware executa as mesmas operações básicas: Inserir, processar e armazenar dados, além de gerar saídas dos dados. Os 5 componentes do computador são: entrada, saída, memória, caminho de dados e controle. Controle e caminho de dados forma o processador.
* Dispositivos de entrada: Mouse e teclado. Dispositivos de saída: Monitor. Existem dispositivos como redes e discos que fornecem tanto entrada quanto saída.

Anatomia do mouse

* Mouse eletromecânico.
* Mouse óptico.

Diante do espelho

* CRT.
* LCD.
* Usando buffer de quadros.

Abrindo o gabinete

* Placa mãe.
* CIs ou chips, dispositivos que combinam dezenas de transistores.
* Memória é onde os programas e dados utilizados são mantidos durante a execução.
* DRAM, acesso randômico. O acesso leva o mesmo tempo independente da localização.
* Memória cache, memória pequena e rápida que age como buffer para a DRAM. Construída com SRAM.
* Arquitetura do conjunto de instruções: É uma interface abstrata entre o hardware e o software que abrange todas as informações necessárias para escrever um programa em linguagem de maquina que será corretamente executado, incluindo instruções, registradores, acesso à memória, E/S e assim por diante.
* ABI, interface binária de aplicação é a combinação do conjunto de instruções básico e a interface do sistema operacional fornecida para os programadores das aplicações.

Um lugar seguro para os dados

* Memória do PC é volátil.
* Memória principal: volátil. DRAMS( Mais rápidas, maior custo).
* Memória secundária: Não-volátil. Discos magnéticos(discos concêntricos cobertos com material de gravação, podem ser achados em interfaces externas. Maior tempo de acesso, por serem mecânicos. Menor custo).
* Discos removíveis: Discos óticos, fita magnética, flash usb, zip.

Comunicação com outros computadores

* Vantagens dos computadores em rede: Comunicação. Compartilhamento de recursos. Acesso remoto.
* LANS conectados através do Ethernet.

Tecnologias para a construção de processadores e memórias.

* Transistor é simplesmente uma chave liga/desliga controlada por eletricidade.
* CI combina vários transistores.
* Lei de Moore: A capacidade em transistores dobra a cada 18 a 24 meses.

Vida real: fabricando chips de Pentium 4.

* Tudo começa com o silício que pode se tornar bom condutor, bom isolante ou uma chave.
* Tudo começa com lingotes de cristal de silício que são fatiados em wafers. Os wafers passam por várias etapas para dopa-los.
* Dies são seções de padrões do wafer.
* Aproveitamento é o número de dies bons pelo total de dies.
* O custo aumenta com o tamanho do die.
* A potência é proporcional ao número de transistores e a frequência de chaveamento.
* Potência = Carga Capacitiva x Voltagem^2 x Frequência de chaveamento.
* Controle do vazamento potencia estática.

**CAPÍTULO 2 - PATTERSON**

2.1 - INTRODUÇÃO

* Instruções são palavras na linguagem do computador.
* Linguagem de máquina são parecidas.
* Conceito de programa armazenado: A idéia de que as instruções e os dados de muitos tipos podem ser armazenados na memória como números, levando ao computador de programa armazenado.

Princípios de projetos

* 1) simplicidade favorece a regularidade
* 2) menor significa mais rápido.
* 3) Agilize os casos mais comuns
* 4) Um bom projeto exige bons compromissos.

2.2 - OPERAÇÕES DO HARDWARE DO COMPUTADOR

* Instruções aritméticas são bastante rígidas.
* Cada instrução é colocada em uma linha.
* Manter fixo o número de operandos simplifica o hardware.

2.3 - Operandos do hardware do computador.

* Operandos precisam ser registrador, por isso são limitados.
* Registradores são primitivas visíveis ao programador.
* Registradores de tamanho de 32 bits.
* Registradores S, de dados.
* Registradores T, temporários.
* Compilador é responsável por associar variáveis a registradores.
* Instrução sub e add.

Operandos em memória

* Estrutura de dados são mantidas em memória.
* Operações aritiméticas só ocorrem nos registradores, por isso é necessário instruções de busca na memória principal.
* Instrução lw

Interface hardware/software

* Restrição de alinhamento: Um requisisto de que os dados estejam alinhados na memória em limites naturais.
* MIPS é big endian.
* Instrução sw.

Constantes ou operandos imediatos

* Usaremos as vezes constantes.

Interface hardware/software

* Spilling registers é o processo de colocar na memória as variáveis menos utilizadas.
* Registradores levam menos tempo para serem acessados e possuem maior vazão.
* Add imediato.

2.4 - Representando instruções

* Números binários.
* Cada parte da instrução é associada a um número diferente e a junção forma uma instrução.
* Todas as instruções possuem 32 bits de largura.

Campos do MIPS

Formato R.

OP -> RS-> RT-> RD-> SHAMT-> FUNCT

OP: opcode 6 BITS

RS: Primeiro operando 5 BITS

RT: Segundo operando 5 BITS

RD: Registrador de destino 5BITS

SHAMT: Shift amount 5 BITS

FUNCT: função. 6 BITS

Formato I

OP->RS->RT->CONST OU ENDEREÇO

OP: OPCODE 6 BITS

RS 5 BITS

RT 5 BITS

CONST 16 BITS

2.5 – Operações lógicas

* Necessidade de atuar sobre bits em uma word.
* Instruções, sll, srl. Sll regDest, regOr, shiftV
* and, andi and regDes, regOr1, RegOr2. Usado para aplicar máscaras.
* or, ori . or regDes, RegOr1, RegOr2
* nor (o not se um dos operandos for 0). Nor regDes, RegOr1, RegOr2
* Nor não possui uma instrução imediata, pois o hardware geralmente o usa para inverter bits.

2.5 – Instruções para tomada de decisões

* Beq R1,R2,local
* Bnq r1,r2,local
* Desvios condicionais, pois necessitam fazer uma verificação antes de realizar o desvio.
* Desvio incondicional j

Interface hardware/software

* Bloco básico: é uma sequencia de instruções sem desvios, exceto, possivelmente no final e sem destinos de desvio exceto no inicio. O compilador executa o desmembramento em blocos básicos.
* Slt, set on less than. Slt r1,r2,r3. Slti existe.
* Não inclui desvios em menor que, por se tratar de algo muito complicado.

Instrução switch/case

* Tabela de endereços de desvio, uma tabela de endereços sequenciais de instruções alternativas.
* Jump register ajuda na tabela.

2.7 – Suporte a procedimentos no hardware do computador

O programa para executar um procedimento precisa seguir seis etapas:

1. Colocar parâmetros em um lugar onde o procedimento possa acessá-los.
2. Transferir o controle para o procedimento.
3. Adquirir os recursos de armazenamento necessários para o procedimento.
4. Realizar a tarefa desejada.
5. Colocar os valores de retorno em um lugar onde o programa principal possa acessá-los.
6. Retornar o controle para um ponto de origem.

* Registradores de argumento a0-a3
* Registradores de retorno de dados: v0 e v1
* Registrador que guarda o endereço de retorno: ra
* Instrução jal para ser usada com procedimentos.
* Registrador pc que guarda a instrução sendo executada (a próxima a ser executada).

Usando mais registradores

* A pilha ajuda na utilização de mais registradores para um procedimento.
* Registrador SP
* Valores temporários são lançados na pilha e depois retirados.

Procedimentos aninhados

* Procedimentos que não chamam outros são os procedimentos folha.

Interface software/hardware

* Registrador gp, global pointer
* É preservado durante a chamada de procedimento: Registradores alvos, stack pointer, registrador de endereço de retorno, pilha acima do sp. Não é preservado: registradores temporários, registradores de argumento, registradores de valor de retorno, pilha abaixo do sp.

Alocando espaço para novos dados na pilha

* Frame de procedimento, ou registro de ativação, é o segmento da pilha contendo os registradores salvos e as variáveis locais de um procedimento.
* Frame pointer, fp.

Alocando espaço para novos dados no heap

* A pilha começa na parte amis alta da memória e cresce para baixo. Dados dinâmicos (heap) e crescem para cima em direção a pilha.

Comunicando-se com as pessoas

* Store byte and load byte, pois caracteres são bytes.

Caracteres e strings em java.

* Unicode e 16bits
* Load halfword e store halfword

2.9 - Endereçamento no MIPS para operados imediatos de 32 bits.

* Formato 7: Jump e jump-and-link

OP. 6 bits

Endereço: 26bits

* Endereçamento relativo ao PC: Desvio que utiliza o PC como base para se somar um offset de desvio. Utiliza offset de word, ao invés de bytes.

Resumo dos modos de endereçamento do MIPS

* Endereçamento em registrador, onde o operando é um registrador.
* Endereçamento de base ou deslocamento: Onde o operando está no local de memória cujo endereço é a soma de um registrador com uma constante na instrução.
* Endereçamento imediato, o operando é uma constante dentro da própria instrução.
* Endereçamento relativo ao PC: Onde o endereçamento é a soma do PC e uma constante na instrução.
* Endereçamento pseudodireito: Onde o endereço do jump são os 26 its da instrução concatenados com o os bits mais altos do PC.

2.10 – Traduzindo e iniciando um programa

Compilador

* O compilador transforma o programa C em assembly.

Montador

* Pseudo-instruções: Variações comuns das instruções em assembly, normalmente tratada como se fosse uma instrução propriamente dita. Ex: move
* O montador transforma o assemly em linguagem de maquina, ou seja, conjunto de instruções e informações para a correta colocação do programa na memória.
* Tabela de símbolos com símbolos e endereços.
* O arquivo objeto contém 6 partes: cabeçalho, segmento de texto, segmento de dados estáticos, informações de realocação, tabela de símbolos, informações de depuração.

Link-editor

* Os procedimentos são compilados independentemente, para evitar gastos de recursos compilando programas que não foram alterados. O linker junta todos os procedimentos de um programa em um arquivo único.
* Coloca os módulos de código e dados simbolicamente na memória. Determina os endereços dos rótulos de dados e instruções. Remenda as referências internas e externas.
* Produz um arquivo executável.

Loader

* Loader: um programa de sistema que coloca programa-objeto na memória principal para que esteja pronto para ser executado.
* Lê o cabeçalho do arquivo para determinar seu tamanho de segmento de texto e dados.
* Cria espaço de endereçamento suficiente.
* Copia instruções e dados do arquivo executável para a memória.
* Copia os parâmetros do programa principal para a pilha.
* Inicializa os registradores da maquina e define os stack pointer.
* Desvia para uma rotina de partida.

DLLs

* Desvantagem das bibliotecas tradicionais de bibliotecas estáticas: As rotinas de bibliotecas fazem parte do executável e não são atualizadas com a atualização da biblioteca.
* Desvantagem: Ela carrega a biblioteca inteira.
* DLLs são bibliotecas que não são link editadas até o programa ser executado.
* Referências são trazidas no próprio arquivo para ajudar o link editor dinâmico durante a execução.
* Link-edição de procedimento tardio onda cada rotina só é link-editada depois de chamada.
* As DLLS exigem espaço para as informações necessárias a link-edição, mas não o fazem em todos os procedimentos de uma biblioteca.

Iniciando um programa em Java

* Programas java são compilados em bytecodes que exigem pouca otimização.
* O programa pode ser escutado em qualquer máquina com o interpretador JVM.
* A interpretação tem a vantagem de ser portátil.
* A interpretação tem a desvantagem do fraco desempenho.
* Compiladores JIT executam os procedimentos enquanto são executados e guardam as instruções de maquina nativas dos procedimentos mais utilizados.

Como os compiladores otimizam

Otimização de alto nível

* São transformações feitas em algo próximo a fonte.
* Utilização de procedimentos inline, substitui uma chamada de função pelo corpo dela.
* Desdobramento de loop, replicar o loop várias vezes no código.

Otimizações locais e globais.

* Otimização local: Atua dentro de um bloco básico, um passo de otimização local geralmente é executado como precursor e sucessor de uma otimização global para limpar código antes e depois da otimização global.
* Otimização global, atua entre vários blocos básicos.
* Alocação de registradores: Aloca variáveis aos registradores para regiões do código. É fundamental para se obter um bom desempenho nos processadores modernos.
* Eliminação de subexpressões comuns: Encontra várias ocorrências da mesma expressão e substitui as subsequentes por uma referência a primeira.
* Propagação de constantes: Propaga o valor de uma constante pelo código.
* Propagação de cópia: Propaga valores que são cópias simples.
* Eliminação de local de armazenamento morto: Elimina locais que não são utilizados.
* Redução de força: Substitui operações complexas por mais simples.
* O compilador prioriza a corretude do código e depois a rapidez.

Otimizações globais de código.

* Movimentação de código: Calcula o valor de um código invariante dentro do loop que ocorre várias vezes de modo a calculá-lo apenas uma.
* Eliminação de variável de indução.

2.17 – Falácias e armadilhas

* Instruções mais poderosas significam maior desempenho: Falácia.
* Escreva em assembly para obter maior desempenho: Falácia.

2.18 – Comentários finais

* Os dois princípios do computador com programa armazenado são o uso de instruções que sejam indistintas de números e o uso de memória alterável para os programas.

**CAPÍTULO 3.3.2 – 3.3.6 – WEBER**

3.3.2 - Unidade Operacional

* Executa as transformações sobre dados especificados pelas instruções de um computador. Compõe-se basicamente de uma unidade lógica e e aritmética, de registradores, e barramentos.
* O porte de um processador é determinado pelo número de registradores e pela quantidade de operações que a ULA pode executar.

ULA

* Realiza as operações aritméticas.
* Operações simples, funções complexas são formadas pela ativação sequencial de operações simples.
* Códigos de condição são indicações que a ULA fornece sobre as operações realizadas.
* Overflow: indica que o resultado da operação não pode ser representado no espaço disponível.
* Sinal: indica se o resultado é negativo ou positivo.
* Carry: Carry out ou borrow-out.
* Zero: indica se o resultado da operação é zero.
* A ula não faz armazenamento.
* A ULA se caracteriza por: comprimento dos operandos, número e tipo de operações, códigos de condição gerados.

Acumulador

* Registrador.
* Armazena um operando e/ou o resultado fornecido pela ULA.
* Controlado por um sinal de carga.
* Caracterizado pelo seu comprimento de bits, a nível de arquitetura.

3.3.3 – Unidade de Controle

* Sinais de controle, fazem o gerenciamento do fluxo interno de dados e o instante preciso em que ocorrem as transferências entre uma unidade e outra.
* Cada sinal de controle comanda uma microoperação: carga em registrador, seleção de dados, ativação da memória, seleção de operação na ula, habilitação de circuito lógico.
* São máquinas de estado finitas.
* Lógica sequencial: os sinais de saída são funções dos sinais de entrada e do estado anterior do sistema.
* Lógica combinacional: os sinais de saída dependem exclusivamente dos sinais de entrada.
* Organização convencional: A unidade de controle é composta por componentes digitais que geram sequencialmente e nos instantes de tempo adequados todos os sinais de controle necessários.
* Organização microprogramada: os sinais de controle estão armazenados numa memória especial. Vários sinais de controle são buscados a cada acesso a memória de controle. Os sinais são agrupados em microinsturções. Um conjunto de microinstruções é um microprograma.
* Depende do registrador de estado e de instruções.

Registradores especiais.

* Apontador de instruções: Manter atualizado o endereço da próxima instrução a ser executada. É o PC. É caracterizado pelo seu comprimento de bits e é função do tamanho da memória.
* Registrador de instruções: Armazena a instrução que está sendo executada. Nele a unidade de controle basear-se-a para determinar quais sinais devem ser ligados. É caracterizado pelo tamanho em bits. O seu tamanho depende do tamanho e codificação das instruções do computador.
* Registrador de estado: Armazena códigos de condição gerados pela ULA. Caracterizado pelo seu comprimento em bits, é função do número de códigos de condição implementados na máquina.

3.3.5 – Conjunto de instruções e modos de endereçamento.

* Uma instrução é uma sequencia de bits devidamente codificados que indica ao computador que sequencia de microoperações ele deve realizar.
* Classificação: Transferencia de ados, aritiméticas e lógicas, teste e desvio.
* Conjunto de instruções são todas as instruções que um computador reconhece e pode executar.
* Modos de endereçamento são as diversas formas em que o endereço do operando, ou endereço de desvio, pode aparecer em um computador.

3.3.6 – Ciclo de busca-decodificação-execução de instrução.

* Tarefas báscias executadas repetidamente.
* Busca: Ler instrução da memória. Copia PC para o registrador de endereços da memória. Ler da memória. Copia do registrador de dados para o registrador de instruções. Atualiza PC.
* Decodificação: Determinação de qual instrução sera executada.
* Execução.

**CAPÍTULO 4 - WEBER**

4.1 – Características

* Largura de endereços e dados de 8 bits
* Dados em complemento de 2
* 1 acumulador de 8 bits
* 1 PC de 8 bits
* 1 registrador de estado com 2 códigos de condição: N e Z

4.2 – Modo de endereçamento

* Enderaçamento direto
* A instrução contém o endereço do operando.

4.3 – Conjunto de instruções do NEANDER

|  |  |  |  |
| --- | --- | --- | --- |
| 0000 | 0 | NOP | Do nothing |
| 0001 | 1 | STA | MEM(end)<-AC |
| 0010 | 2 | LDA | AC<-MEM(end) |
| 0011 | 3 | ADD | AC<-AC+Mem(end) |
| 0100 | 4 | OR | AC<-AC or MEM(end) |
| 0101 | 5 | AND | AC<-AC and MEM(end) |
| 0110 | 6 | NOT | AC<-not AC |
| 1000 | 8 | JMP | PC<-End |
| 1001 | 9 | JZ | If Z = 1 PC<-end |
| 1010 | A | JN | If N=1 PC<-end |
| 1111 | F | HLT | Stop |

4.4 – Códigos de Condição

* N e Z saem da ULA e LDA

4.5 – Formato da instrução

* Bits 7-4 = Código da instrução
* Bits 3-0 = Don’t care
* 8 bits para o operando.