Trabalho final AOC 2 - Comparação entre arquiteturas x86 e MIPS

João Gabriel Bernardes de Oliveira

Ciencia da Computação - Poços de Caldas – MG

{gabriel, joão}jgbbernardes5@gmail.com

Resumo. *Este trabalho tem como objetivo realizar uma análise comparativa entre as arquiteturas de computadores x86 e MIPS.*

1. Introdução as Arquiteturas

As arquiteturas de computador x86 e MIPS representam duas abordagens distintas para o design de processadores. O processador Intel x86 usa uma arquitetura Cisc (computador conjunto de instruções) complexa, o que significa que há um número modesto de registros de finalidade especial em vez de grandes quantidades de registros de uso geral. Isso também significa que instruções complexas de finalidade especial serão predominadas. Essa arquitetura x86 aparece primeiro como o Intel 8086 CPU lançado em 1978. Já o MIPS é uma Arquitetura de Conjunto de Instruções (Instruction Set Architecture – ISA), desenvolvida pela empresa MIPS Computer Systems, que hoje é chamada de MIPS Technologies. A empresa foi fundada em 1984 por um grupo de pesquisadores da Universidade de Stanford e o foco era os microprocessadores com Arquitetura RISC.

1.1 Hardware das Arquiteturas x86 e MIPS

A arquitetura x86 é amplamente utilizada em uma variedade de processadores fabricados por diferentes empresas. Como por exemplo a Intel Xeon que usa processadores projetados para servidores e estações de trabalho, otimizados para cargas de trabalho intensivas e aplicações empresariais. AMD EPYC: Processadores de servidor de alta performance, competindo diretamente com a linha Intel Xeon em data centers e ambientes empresariais. Cada uma dessas linhas de processadores x86 tem características específicas voltadas para diferentes segmentos de mercado, desde consumidores domésticos até data centers empresariais, demonstrando a flexibilidade e adaptação da arquitetura x86. A arquitetura MIPS é conhecida por sua simplicidade e eficiência, especialmente em sistemas embarcados e dispositivos específicos, alguns exemplos de aplicações são MIPS32 e MIPS64, processadores que abrangem várias gerações, otimizadas para diferentes aplicações, desde sistemas embarcados até roteadores e dispositivos de rede, A arquitetura MIPS também é encontrada em muitos dispositivos específicos, como roteadores de rede, sistemas de controle industrial, equipamentos de comunicação, sistemas de áudio e vídeo, entre outros.

1.2 Conjuntos de Registradores utilizados por cada Arquitetura

A arquitetura x86 consiste nos seguintes registros inteiros sem privilégios.

Arquitetura x86:

Eax	Acumulador
Ebx	Registro base
ecx	Registro de contador
Edx	Registro de dados – pode ser usado para acesso à porta de E/S e funções aritméticas
Esi	Registro de índice de origem
Edi	Registro de índice de destino
Ebp	Registro de ponteiro base
Esp	Ponteiro de pilha

Arquitetura MIPS:

Nome do Registrador	**Número**	**Binário**	**Uso**
$zero	0	000 000	constante zero
$at	1	000 001	reservado para o montador
$v0	2	000 010	avaliação de expressão e resultados de uma função
$v1	3	000 011	avaliação de expressão e resultados de uma função
$a0	4	000 100	argumento 1 (passam argumentos para as rotinas)
$a1	5	000 101	argumento 2

Esses são apenas alguns exemplos de registradores entre MIPS e x86, podemos perceber diferenças, enquanto a arquitetura x86 oferece uma gama mais ampla e complexa de registradores para suportar uma variedade de aplicações e cenários de software, a arquitetura MIPS se destaca pela sua simplicidade e uniformidade, favorecendo eficiência e economia de recursos em sistemas embarcados e integrados. A escolha entre essas arquiteturas muitas vezes depende das necessidades específicas do projeto, incluindo desempenho, consumo de energia, e compatibilidade com o ecossistema de software existente.

2. Comparação de Instruções x86 e MIPS

No processador **x86**, as instruções são de tamanho variável, portanto, desmontar para trás é um exercício de correspondência de padrões. Para desmontar para trás de um endereço, você deve começar a desmontar em um ponto mais atrás do que realmente deseja ir e, em seguida, olhar para a frente até que as instruções comecem a fazer sentido. As primeiras instruções podem não fazer sentido porque você pode ter começado a desmontar no meio de uma instrução. Infelizmente, há a possibilidade de que a desmontagem nunca seja sincronizada com o fluxo de instruções e você terá que tentar desmontar em um ponto de partida diferente até encontrar um ponto de partida que funcione. A notação geral para instruções é colocar o registro de destino à esquerda e a origem à direita.

Para economizar espaço, muitas das instruções são expressas em formato combinado como por exemplo:

*	MOV	**r1**, **r**/m/#n	**r1** = **r**/m/#n

significa que o primeiro parâmetro deve ser um registro, mas o segundo pode ser um registro, uma referência de memória ou um valor imediato.

Exemplo de aplicação para memória, transferência de Dados e conversão de Dados:

as instruções de transferência de dados e memória não afetam os sinalizadores.

Endereço efetivo

*	LEA	**r**, m	Carregar endereço efetivo. (r = endereço de m)

Por exemplo, LEA eax, [esi+4] significa eax = esi + 4. Essa instrução geralmente é usada para executar aritmética.

Já no processador **MIPS**

Program em C	Assembly MIPS
a = b + c; d = a – c	add a,b,c sub d,a,c

f = (g + h) – (i + j);	add t0,g,h
	add t1,i,j
	sub f,t0,t1
	o compilador cria t0 e t1 .

No MIPS os operandos das instruções são registradores, mas expecificamente 32 registradores de 32 bits.

Exemplo de instruções de movimentação de dados:

Load e Store – lw : instrução de movimentação de dados da memória para registrador (load word) – sw: instrução de movimentação de dados do registrador para a memória (store word).

Tabela Comparação das duas arquiteturas:

Categoria	x86	MIPS
Arquitetura	CISC	RISC
Tamanho da Instrução	Variável (1 a 15 bytes)	Fixo (32 bits)
Número de Instruções	Grande, muitas instruções complexas	Pequeno, instruções simples
Registro de Uso	Poucos registradores gerais	Muitos registradores de propósito geral
Modos de Endereçamento	Complexos e variados	Simples (direto, imediato, base+offset)
Execução de Instruções	Alta variabilidade em ciclos por instrução	Foco em uma instrução por ciclo
Popularidade/Aplicação	Desktops, laptops, servidores	Sistemas embarcados, dispositivos móveis

3. Análise de Código Assembly:

1º código em Linguagem C:

```
#include <stdio.h>
int main() {
int a, b;
printf("Digite dois números: ");
scanf("%d %d", &a, &b);
printf("A soma é: %d\n", a + b);
return 0;
```

Codigo em x86 :

```
.section .data
format:
    .string "Digite dois números: "
    len:
    .int 21

sum_format:
    .string "A soma é: %d\n"
    sum_len:
    .int 13

.section .bss
a:
    .space 4
b:
```

Codigo em MIPS:

```
# Código em assembly x86 para calcular a soma de dois números

# Incluindo a biblioteca stdio.h
.section .data
format:
    .string "Digite dois números: "
    len:
    .int 21

sum_format:
    .string "A soma é: %d\n"
    sum_len:
    .int 13

.section .bss
a:
    .space 4
b:
    .space 4

.section .text
.globl main
```

```
    .space 4

.section .text
.globl main
main:

    movl $4, %eax
    movl $1, %ebx
    movl $format, %ecx
    movl $len, %edx
    int $0x80



    movl $3, %eax
    movl $0, %ebx
    leal a, %ecx
    movl $4, %edx
    int $0x80

    movl $3, %eax
    movl $0, %ebx
    leal b, %ecx
    movl $4, %edx
    int $0x80



    movl a, %eax
    movl b, %ebx
    addl %ebx, %eax

    movl $4, %eax
    movl $1, %ebx
    movl $sum_format, %ecx
    movl $sum_len, %edx
    int $0x80

    movl %eax, %ebx
```

```
main:
    # Chamada para imprimir "Digite dois números: "
    movl $4, %eax
    movl $1, %ebx
    movl $format, %ecx
    movl $len, %edx
    int $0x80

    # Leitura dos números digitados pelo usuário
    movl $3, %eax
    movl $0, %ebx
    leal a, %ecx
    movl $4, %edx
    int $0x80

    movl $3, %eax
    movl $0, %ebx
    leal b, %ecx
    movl $4, %edx
    int $0x80

    # Cálculo da soma
    movl a, %eax
    movl b, %ebx
    addl %ebx, %eax

    # Chamada para imprimir "A soma é: " e o resultado
    movl $4, %eax
    movl $1, %ebx
    movl $sum_format, %ecx
    movl $sum_len, %edx
    int $0x80

    movl %eax, %ebx
    movl $1, %eax
```

movl $1, %eax int $0x80 movl $1, %eax xorl %ebx, %ebx int $0x80	int $0x80 # Retorno da função main movl $1, %eax xorl %ebx, %ebx int $0x80

4. Processadores Hibridos, CISC e RISC

CISC:

Arquiteturas CISC, como a x86, usam instruções complexas e especializadas. Elas conseguem fazer mais com menos código, o que simplifica a programação. Por exemplo, uma única instrução CISC pode realizar várias tarefas ao mesmo tempo, como acessar memória, fazer cálculos e processar informações. Isso pode ser rápido, mas às vezes leva mais tempo para completar uma instrução. Em geral, processadores CISC são rápidos, mas consomem mais energia do que os RISC. Isso é algo importante de considerar ao escolher qual arquitetura usar para diferentes tipos de computação.

RISC:

RISC é uma arquitetura de processador que utiliza um conjunto simplificado de instruções simples e otimizadas. Ao contrário do CISC, que favorece instruções complexas e especializadas, o RISC foca na execução rápida e eficiente de instruções. Essa abordagem permite que processadores RISC, como a arquitetura Arm, lidem com muitas instruções em um período mais curto. Embora softwares compilados para RISC possam requerer mais código de baixo nível, como Assembly, devido às suas instruções mais simples, o RISC oferece vantagens como gerenciamento de memória mais simples e execução previsível. Em geral, as arquiteturas RISC priorizam a eficiência energética, mas podem apresentar desempenho inferior em operações complexas em comparação com processadores CISC.

Conclusão:

A comparação entre as arquiteturas x86 e MIPS revela que cada uma é ideal para diferentes aplicações. A arquitetura x86, com seu conjunto de instruções complexo (CISC), é potente e flexível, sendo comum em PCs e servidores. A MIPS, com um conjunto de instruções reduzido (RISC), é eficiente e de baixo consumo, ideal para dispositivos embarcados. O Assembly MIPS é mais simples, facilitando o aprendizado, enquanto o x86 oferece maior controle e desempenho. Processadores híbridos combinam as melhores características de ambas as arquiteturas, atendendo a diversas demandas da computação moderna.

5. Referencias

https://www.ic.unicamp.br/~pannain/mc542/aulas/ch3_arq.pdf

https://learn.microsoft.com/pt-br/windows-hardware/drivers/debugger/x86-architecture

https://tecnoblog.net/responde/qual-e-a-diferenca-entre-arquitetura-risc-e-cisc-processador/