	Laboratório de Arquitetura de Computadores	Data:05/11/2019
	Relatório de Experimentos	Módulo: 2

Nome: Giovanne Prestes Dias

RA 171029

Título: Arquitetura x86 – Assembly PC.

Objetivos:

- Adquirir conhecimentos em arquitetura de computadores;
- Uso do ambiente do Visual Studio;
- Instruções básicas usando linguagem Assembly;
- Manipulação de dados em registradores e memória.

Material Utilizado:

- Software Visual Studio.

Relatório:

Rodar os exemplos apresentados;

Analisar e comentar os códigos em Assembly gerados pelos 5 exemplos;

Pesquisar o significado da seguinte diretiva: `DWORD PTR _x$[ebp]`

Criar um programa em Assembly para indicar a posição do maior valor em uma matriz com 10 elementos

1) Introdução com a apresentação dos assuntos abordados no experimento;

A linguagem Assembly é uma linguagem de baixo nível constituída por um conjunto de mnemônicas e abreviações. Em comparação com código máquina (uma série de números em formato binário), Assembly torna as instruções mais fáceis de lembrar, facilitando a vida ao programador.

O uso da linguagem Assembly já data da década de 1950, sendo nessa altura uma linguagem bastante popular. Atualmente, com a evolução das linguagens de alto nível, é usada majoritariamente no desenvolvimento de drivers, sistemas integrados e na área de reverse engineering (como a maior parte dos programas só estão disponíveis num executável binário ou código máquina, é muito mais fácil traduzi-los para linguagem Assembly do que para linguagens de alto nível—este processo designa-se por disassembly).

O código fonte de um programa em linguagem Assembly está diretamente relacionado com a arquitetura específica do processador alvo—ao contrário das linguagens de alto nível, que são geralmente independentes da plataforma, bastando recompilar o código para o executar numa arquitetura diferente.

A linguagem Assembly é traduzida para código máquina através de um programa chamado assembler. Um assembler é diferente de um compilador na medida em que traduz as mnemônicas uma-a-uma para instruções em código máquina, enquanto um compilador traduz as instruções por blocos de código.

Os 8 GPRs, ou Registradores de Uso Geral, são os seguintes (por ordem de introdução na pilha ao executar a instrução `PUSHAD`):

- EAX - Acumulador. Usado em operações aritméticas.
- ECX - Contador. Usado em loops.

- EDX - Registrador de Dados. Usado em operações de entrada/saída e em multiplicações e divisões. É também uma extensão do Acumulador.
- EBX - Base. Usado para apontar para dados no segmento DS.
- ESP - Apontador da Pilha (Stack Pointer). Aponta para o topo da pilha (endereço mais baixo dos elementos da pilha).
- EBP - Apontador da base do frame. Usado para acessar argumentos de procedimentos passados pela pilha.
- ESI - Índice da fonte de dados a copiar (Source Index). Aponta para dados a copiar para DS:EDI.
- EDI - Índice do destino de dados a copiar (Destination Index). Aponta para o destino dos dados a copiar de DS:ESI.

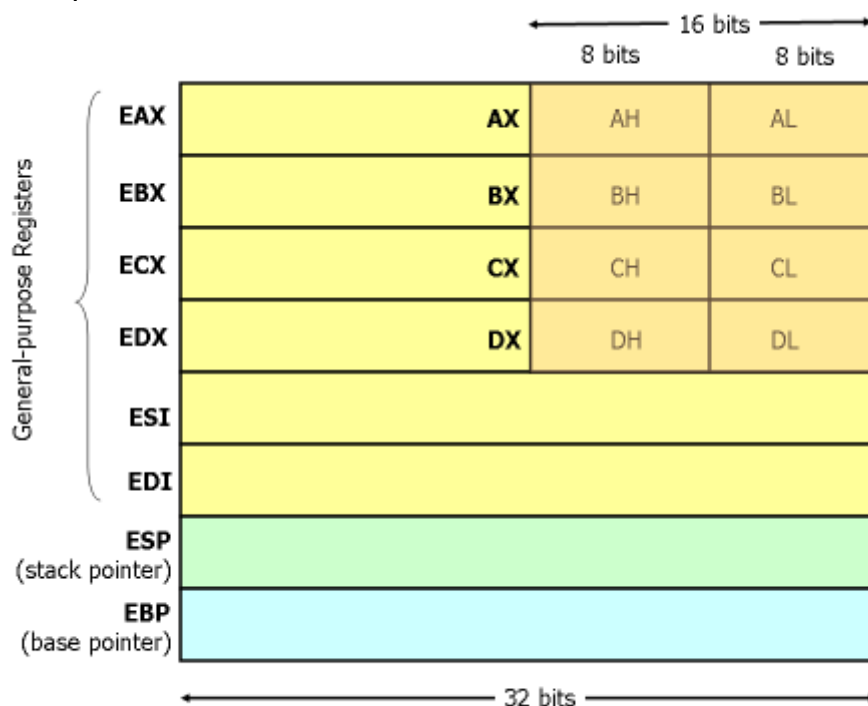


Figure 1. x86 Registers

A seguinte diretiva: `DWORD PTR _x$[ebp]`, significa que é um elemento com tamanho de 32 bits.

2) Procedimento experimental executado (quais foram os passos executados para a criação do projeto);

Primeiramente, criamos um novo projeto no Visual Studio, selecionei em Aplicativo de Console (em C++) e colocamos um nome para o projeto.

Após abrir o projeto e rodar a primeira vez, para ver se estava tudo ok, chegamos neste código:

```
void Maior(void) {
    int v[10] = { 3, 2, 4, 7, 12, 20, 133, 11, 9, 144 }, maior, indice = 4, inteiro = 4, cont = 0;

    __asm {
        mov eax, DWORD PTR[v]
        mov[maior], eax

        INICIO :
            mov ecx, indice
            mov eax, [maior]
            cmp eax, DWORD PTR[v + ecx]
            jl MAIORTTRUE
            jg MAIORFALSE

        MAIORTTRUE :
            mov eax, DWORD PTR[v + ecx]
            mov[maior], eax
        MAIORFALSE :
            mov ecx, [indice]
            add ecx, [inteiro]
            mov[indice], ecx
            inc cont
            cmp cont, 10
            jl INICIO
            jmp FIM


        FIM :
    }
}
```

3) Análise dos resultados (acrescentar comentários sobre o funcionamento do projeto);

Primeiramente, atribuímos valores para o vetor de 10 elementos, para que fosse possível fazer o teste, declaramos as variáveis para guardar o maior valor ("maior"), outra para guardar o valor do índice que irá percorrer o vetor ("índice"), para guardar o valor em bytes de cada elemento do vetor, como todos são inteiros, todos ocupam 4 bytes ("inteiro") e uma variável para contar até 10, para que fosse possível percorrer todos os 10 elementos ("cont").

Após a declaração de variáveis, atribuímos o primeiro valor do vetor para a variável "maior", para que fosse possível a comparação com os demais elementos. E com isso começamos o programa, onde mandamos a variável "índice" para o "ecx", e a variável "maior" para eax para fazer a comparação da maior (até o momento) com o próximo elemento do vetor.

Caso o valor do próximo elemento seja menor do que o valor que está contido na variável "maior", só incrementamos o índice e o contador e verificamos se o contador já

	Laboratório de Arquitetura de Computadores	Data:05/11/2019
	Relatório de Experimentos	Módulo: 2

chegou no valor 10, se chegou acaba o programa, caso não tenha chego ainda, voltamos para o início, para que seja feita a comparação novamente com o próximo elemento.

Se o elemento for maior, colocamos o valor do próximo elemento do vetor na variável maior e fazemos o mesmo passo de caso ele for menor. Tendo assim no final do programa, o maior valor do vetor na variável “maior”

4) Conclusão.

Com este experimento obtivemos conhecimento da diretiva DWORD PTR _x\$[ebp], e também mais conhecimento com a arquitetura de computadores, com o uso do Visual Studio, também com a linguagem Assembly e manipulação em registradores e memória na mesma.

Assim, concluimos que o programa mostrado acima, foi concluído conforme o solicitado para este experimento, onde guarda na variável “maior” o maior elemento do vetor de 10 posições