Universidade Federal do Espirito Santo

Departamento de Engenharia

Engenharia de Computação

Laboratório 02
Codificação básica de instruções

Aluno(a): Arthur Coelho Estevão

Professora: Camilo Arturo Rodriguez Diaz

Vitória

2022

1 Introdução

Este Laboratório tem como objetivo principal Verificar a codificação básica das instruções dos microprocessadores 8086/8088 e os seus modos de endereçamento da memória.

2 Teoria

2.1 Registradores de segmento da CPU

Os registradores de segmento da CPU armazenam os endereços iniciais de memória onde serão carregados:

- Programa (CS)
- Declaração de variáveis (DS)
- Pilha (SS)
- Extra (ES)(por exemplo: outra RAM, heap ou usado para apontar para qualquer região da RAM do sistema sem precisar de mudar o conteúdo de CS, DS, SS)

A organização dos segmentos de dados pode ser vista em Figura 1.

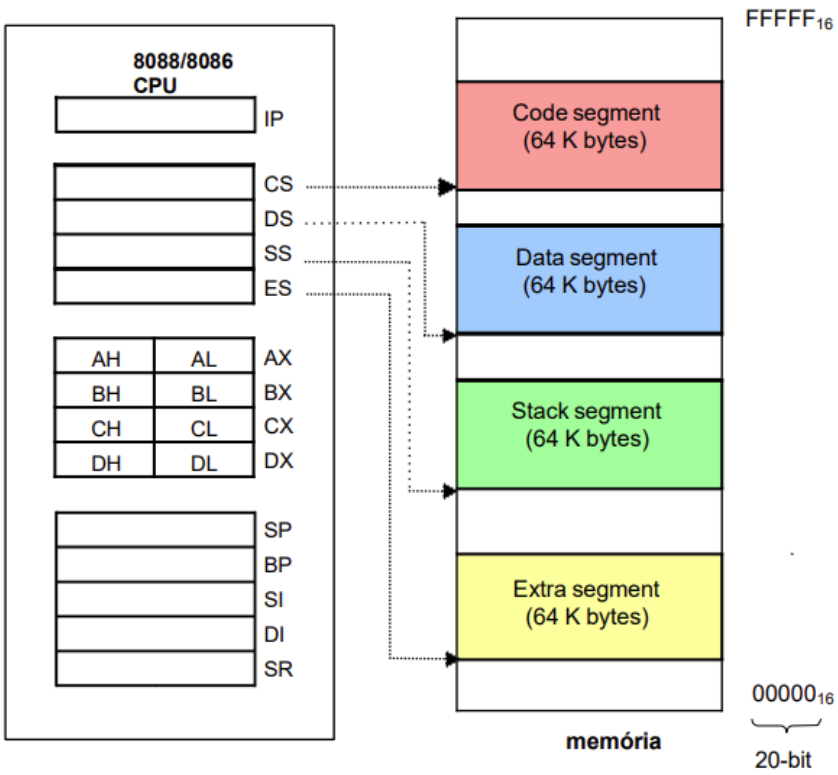

Figura 1: Memoria

2.2 Estrutura das Instruções

A estrutura é com 6 bytes (48 bits no total), sendo cada byte responsável por algo em específico. A contagem é da esquerda pra direita.

- Byte 1 = opcode
- Byte 2 = mod reg rm
- Byte 3 = lo desloc (não se preocupe)
- Byte 4 = hi desloc (não esquenta)
- Byte 5 = lo data (ignore)
- Byte 6 = hi data (deixa quieto, não é importante agora)

2.3 Byte 1 - opcode, em geral

Como cada byte tem 8 bits, o primeiro byte tem uma especificação em 8 bits. Do bit 7 ao 2 (6 bits) responsável pelo ”código da operação”, também chamado de ’opcode’. Quando o processador for executar esse código, ele vai saber qual operação realizar (soma, subtração, mov, etc)

Os outros bits são responsáveis por dizer algumas coisas sobre os ’argumentos’ das instruções. Serve pra dizer se o ’ax’ em ’mov ax,[bx]’ é um registrador que vai receber um valor ou vai enviar um valor.

- Bit 1 = D (destino)
- Bit 0 = W (word, palavra)

Se D = 0 o registrador especificado no campo ”REG”será a fonte de informação (se ele vai ficar no lado direito do ’mov destino,fonte) Caso contrário o registrador especificado no campo ”REG”será para a onde a informação destinará (se ele vai ficar no lado esquerdo do ’mov destino,fonte’)

Se W = 0 é uma instrução está mexendo com bytes (AH, AL, que possuem 8 bits) caso contrario é uma instrução está mexendo com words (AX, BX, DX, que possuem 16 bits).

2.4 Byte 2 - mod, reg, r/m

O mod é o ”modo”(jeito ou maneira de realizar a ação) que você realiza as instruções. Se você for usar uma instrução que lida entre registradores, o ’mod’ é um valor. Se você usar instruções que pegam informação de registrador para jogar na memória RAM, é outro, e assim vai. ’reg’ é o registrador que define se vai ser fonte ou destino, mencionado anteriormente, já r/m é o meio que ”de registrador vai memória (salvar/save)”ou ”da memória vai para o registrador (carregar/load)”.

3 Resultados

A primeiras operações executadas no laboratório foram elaboradas para a organização dos segmentos da CPU, que inicialmente são predefinidos com o mesmo seguimento **075F**. A partir disso foram modificados CS, DS, SS e ES para que apontem para parágrafos adjacentes a partir do segmento CS= 075F com uma Diferença de 16 bytes na memória, Figura 2, por meio da utilização do comando R visto no ultimo laboratório.

Figura 2: Organizando registradores de segmento

Modo de Endereçamento	Exemplo	Segmento
Registro indireto	**MOV AX, [BX]**	**DS**
	MOV AX, [BP]	**SS**
	MOV AX, [SI]	**DS**
	MOV AX, [DI]	**DS**
De Base	**MOV AX,[BX+DESL]**	**DS**
	MOV AX,[BP+DESL]	**SS**
Direto	**MOV AX,[1000]**	**DS**
Indexado	**MOV AX,[SI+DESL]**	**DS**
	MOV AX,[DI+DESL]	**DS**
De Base Indexado	**MOV AX,[BX+SI+DESL]**	**DS**
	MOV AX,[BX+DI+DESL]	**DS**
	MOV AX,[BP+SI+DESL]	**SS**
	MOV AX,[BP+DI+DESL]	**SS**

Tabela 1: Resumo dos valores medidos em cada cenários no primeiro caso de teste

Na Figura 3 é utilizado o comando U desassemble, que torna possível verificar os bytes gerados para cada instrução e comparar com os campos definidos no formato da instrução típica Figura 4.

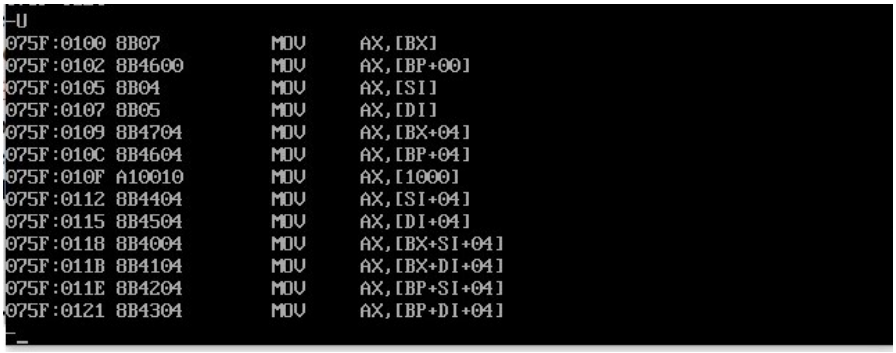

Figura 3: Desassemble comandos

Ao escrever a instrução do tipo **MOV** o montador pode gerar diversas formas de codificação das instruções de máquina, dependendo dos registros envolvidos e do modo de endereçamento utilizado na instrução, O formato de uma instrução típica do 8086/8088 pode ser é de 6 bytes sendo que os bytes 3 à 6 podem existir ou não dependendo do tipo da instrução, no caso do **MOV** não existem, restando apenas o byte 1 e 2, a estrutura pode ser analisada na secção 2.

Pegando o {**MOV AX, [BX]**} como exemplo, o datasheet nos da que o opcode de **MOV** e 1000'10DW, onde D e W já foram especificado na secção 2, Como AX é o destino e é um REG estamos mexendo com WORDS, então D = 1 e W = 1 portanto o primeiro byte vai ser 1000'1011, que em Hexa, é 0x8B, já o segundo byte é encontrado pelo mod = 00 pois esta em modo memória sem deslocamento como W = 1, o REG tem que ser 000 para que pois estamos lidado com AX. A fonte, no caso, r/m, é [BX]. Para o caso de mod = 00, vemos que r/m tem que ser 111 para dar [BX] como fonte, portanto o byte 2 fica 0000'0111, que em hexa é 0x07, todos os outros comandos podem ser encontrados na Figura 4.

	Byte 1					Byte 2				
	Código de operação	W	D	Binario	Hexa	MOD	REG	R/M	Binario	Hexa
MOV AX,[BX]	100010	1	1	10001011	8B	00	000	111	00000111	07
MOV AX,[BP+00]						01		110	01000110	46 00
MOV AX,[SI]						00		100	00000100	04
MOV AX,[DI]						00		101	00000101	05
MOV AX,[BX+04]	100010	1	1	10001011	8B	01	000	111	01000111	47 04
MOV AX,[BP+04]						01		110	01000110	46 04
MOV AX,[1000]	101000	0	1	10010001	A1	00	000	000	00000000	00 10
MOV AX,[SI+04]	100010	1	1	10001011	8B	01	000	100	01000100	44 04
MOV AX,[DI+04]						01		101	01000101	45 04
MOV AX,[BX+SI+04]	100010	1	1	10001011	8B	01	000	000	01000000	40 04
MOV AX,[BX+DI+04]						01		001	01000001	41 04
MOV AX,[BP+SI+04]						01		010	01000010	42 04
MOV AX,[BP+DI+04]						01		011	01000011	43 04

W	0	byte
	1	word
D	0	Fonte REG
	1	Destino REG

Figura 4: Formato das instruções

4 Referências

https://ava.ufes.br/pluginfile.php/539919/mod$_{r}esource/content/2/Modelo$