EA870 - Laboratório de Computação

Laboratório 1
Arquitetura de Computadores - Revisão
Introdução ao SBC68K
Introdução ao Motorola MC68000
Introdução ao Firmware TUTOR

1. Introdução

As experiências e programas a serem desenvolvidos no decorrer deste curso utilizam como plataforma hardware o SBC68K (Single Board Computer baseado no micro-processador Motorola MC68000). O objetivo desta disciplina é o aprendizado/fixação das características organizacionais básicas de um micro-processador, utilizando como objeto de experimentação o micro-processador MC68000.

Este roteiro está organizado como se segue:

- Seção 2: Breve revisão da arquitetura típica de um computador digital
- Seção 3 : Arquitetura do SBC68K
- Seção 4: Micro-processador MC68000
- Seção 5: TUTOR - ferramenta de auxílio ao desenvolvimento de programas
- Seçao 6: Programa Terminal - ferramenta para a comunicação PC - SBC68K
- Seção 7: Exercícios.

2. Arquitetura de Computadores - Revisão

Um computador digital é constituído tipicamente de:

a) Unidade Central de Processamento (UCP ou CPU)
b) Memória
c) Interfaces de Entrada e Saída

A figura 1.1 apresenta a arquitetura típica de um computador digital (máquina de von Neumann).

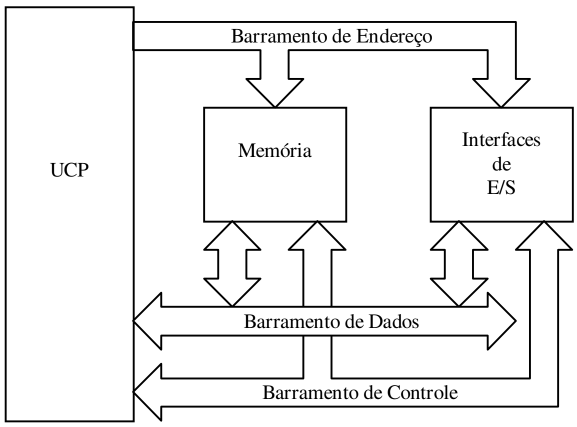

Fig. 1.1: Arquitetura típica de um computador.

Em geral, a **memória** é um arranjo linear de células individuais de armazenamento, sendo que cada célula possui um **endereço**, que permite a sua identificação individual, e um **conteúdo**, que é a informação armazenada na célula. A memória pode ser utilizada para armazenar **instruções**, que são os elementos de informação que comandam as atividades da UCP, ou **dados**, que são os elementos de informação que são processados e/ou gerados pela UCP. Um conjunto de instruções logicamente relacionadas armazenado na memória é denominado de **programa**.

As **interfaces de entrada e saída** permitem a troca de informações entre o computador e o mundo exterior, adequando elétrica e temporalmente os sinais que fluem entre estes dois mundos. Exemplos de dispositivos externos que podem ser acoplados a estas interfaces são: monitor, teclado, disco flexível, "mouse", leitora de cartão, disco rígido, etc.

A **UCP** é o componente ativo do sistema, que controla a função dos outros elementos através dos barramentos de endereço, dados e controle. Para desempenhar o seu papel controlador, a UCP tem que ser capaz de ler instruções da memória, decodificá-las e executá-las. A UCP deve ser ainda capaz de efetuar acessos adicionais à memória ou mesmo às interfaces de E/S, para buscar operandos ou guardar resultados que eventualmente se façam necessários durante a execução de uma instrução. Pelo **barramento de endereço**, a UCP especifica o elemento (célula de memória ou interfaces de E/S) com quem ela deseja se comunicar. Pelo **barramento de controle**, a UCP define a operação a ser realizada (leitura ou escrita). Pelo **barramento de dados**, a UCP envia (operação de escrita) ou recebe (operação de leitura) informação.

Tipicamente uma UCP é constituída de:

a) Registradores
b) Unidade de Aritmética e Lógica (UAL ou ALU)
c) Circuitaria de controle

Registradores são células de armazenamento internas à UCP. Geralmente, uma UCP possui um conjunto de registradores de propósito geral, utilizados para armazenamento temporário, e registradores reservados para uso dedicado. Dentre estes últimos, destaca-se: o registrador contador de programa (program counter - PC), registrador apontador de pilha, e registrador de estado.

A UCP trabalha de modo cíclico: A UCP lê da memória a próxima instrução a ser executada (este acesso de leitura é denominado de busca de instrução), executa a operação especificada pela instrução, busca a próxima instrução, executa, e assim sucessivamente. O **registrador contador de programa**, ou simplemente **PC**, é o elemento registrador que indica o endereço da próxima instrução a ser executada. A UCP utiliza o conteúdo do PC para acessar a célula de memória onde reside a próxima instrução a ser executada. Após a leitura de uma instrução, o conteúdo do PC é automaticamente incrementado pela UCP, para fazê-lo apontar para a próxima instrução a ser executada. Assim, de um modo geral, as instruções que compõem um programa são armazenadas sequencialmente na memória, em ordem crescente de endereçamento - endereços menores contêm as primeiras instruções. A única forma de um programador alterar esta execução sequencial de instruções é fornecida pelas instrucões que alteram explicitamente o valor do PC, tais como: instruções de salto (e.g, jump, branch, etc.) e de chamada à subrotina.

Em uma **instrução de salto** (ou instrução de desvio) pode-se especificar o endereço da próxima instrução a ser executada, de forma a permitir a transferência do fluxo de execução para a sequência de instruções residente em outro trecho de memória. Da mesma forma que qualquer outra instrução, após a busca de uma instrução de salto, a UCP incrementa automaticamente o conteúdo do PC. Entretanto, diferentemente das outras instruções, a execução da instrução de salto altera o valor do PC, carregando-o com o endereço especificado na instrução. Após a execução da instrução de salto, a UCP busca, como sempre, a instrução apontada pelo PC, continuando o ciclo de processamento, que, neste caso, resulta em um salto para outra região da memória.

De modo análogo às intruções de salto, uma **instrução de chamada à subrotina** permite a especificação do endereço da próxima instrução. Diferentemente entretanto, a execução de uma instrução de chamada à **subrotina** ocasiona adicionalmente o empilhamento do endereço de retorno na estrutura de pilha, cujo topo é apontado pelo **registrador apontador de pilha**. Uma **subrotina** sempre deve ser terminada por uma instrução de retorno, que retira o endereço de retorno da pilha e o coloca no PC.

O **registrador de estado** (ou registrador de condições) é geralmente interpretado bit-a-bit. Os bits podem estar associados a resultados de operações aritméticas e lógicas, como, por exemplo: se o resultado é zero, se o resultado é negativo, se houve overflow, etc. Processadores diferentes geralmente apresentam maneiras diferentes de acionar os bits do registrador de estado. É importante consultar o jogo de instruções do processador para saber como cada instrução afeta os bits do registrador de estado.

3. O Micro-Computador SBC68K

Neste laboratório como plataforma de experimentação será utilizado o micro-computador SBC68K baseado no micro-processador MC68000. Esta máquina, observada a organização apresentada na Fig. 1.1, é composta de:

- **UCP:**
  Micro-processador MC68000 localizado no soquete U22 da placa.
- **Memória:**
  RAM (random access memory) localizada nos soquetes U29 e U30;
  EPROM (eraseable programmable read only memory) localizada nos soquetes U27 e U28.
- **Interfaces de Entrada e Saída:**
  Interface Serial DUART (dual asynchronous receiver/transmitter) MC68681 localizada no soquete U25;
  Interface Paralela PI/T (parallel interface/timer) MC 68230 localizada no soquete U31;
  Interface controladora de disco WD1772 localizada no soquete U19.

Uma descrição detalhada do hardware é apresentada em [3]- capítulos 1 e 5. As interfaces de entrada e saída serão tratadas em maiores detalhes a partir do laboratório 3.

4. O Micro-processador MC68000

4.1 Introdução

A programação a nível de máquina de qualquer computador começa pelo aprendizado:

1) da organização dos registradores internos ao processador,
2) da organização da memória e forma de acesso a esta, e naturalmente
3) do jogo de instruções suportado pelo processador.

O MC68000 é um micro-processador de 16 bits (tamanho do barramento de dados), que suporta operações com dados de 8 bits (byte), 16 bits (palavra ou word), ou 32 bits (palavra longa ou long word).

4.2 Organização dos Registradores

A figura 2.1 apresenta os registradores internos do micro-processador MC68000.

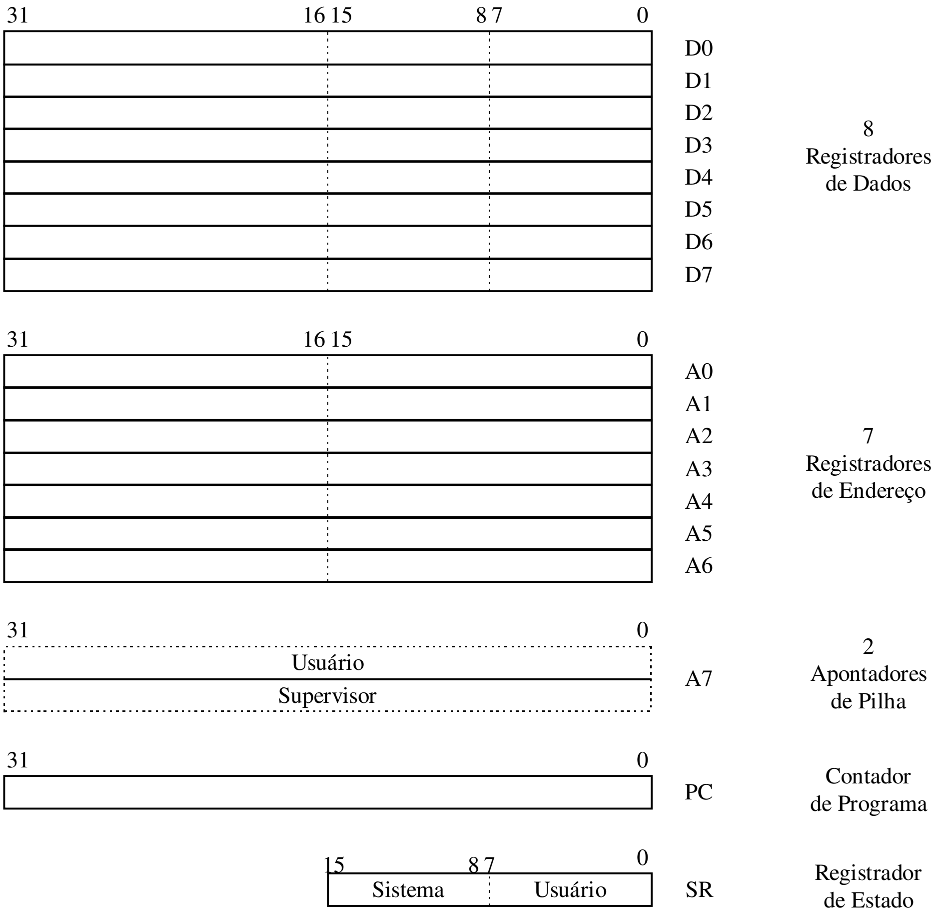

Fig. 2.1: Registradores do micro-processador MC68000.

Registradores de dados: Os 8 Registradores de Dados (D0 - D7), com 32 bits cada, suportam operandos de tipo byte, word e long word. Operandos do tipo byte ocupam os 8 bits menos significativos do registrador de dados; operandos do tipo word ocupam os 16 bits menos significativos e os long word todos os 32 bits do registrador. O bit menos significativo é endereçado como bit 0; o bit mais significativo é endereçado como bit 31. Uma transferência de dados para/de um destes registradores afeta/utiliza apenas a porção da parte menos significativa que for apropriada; a parte mais significativa não é afetada/utilizada.

Registradores de endereço: Os 7 registradores de endereço (A0 - A6) e o apontador de pilha (A7) comportam 32 bits cada. Tais registradores suportam apenas operandos do tipo word e long word. Ao servir como operando fonte, apenas a porção apropriada do registrador é utilizada - os 16 bits menos significativos (word), ou todos os 32 bits (long word). Porém, um registrador de endereço ao ser utilizado como operando destino terá sempre todos os seus 32 bits afetados independentemente do tamanho da operação efetuada (word ou long word). Para operações de tamanho word que envolvam um registrador de endereço como operando destino, tem-se sempre, antes da execução da operação, a extensão do operando fonte para 32 bits, mantendo-se o sinal (sign-extended).

Adicionalmete, tem-se ainda o **registrador contador de programa** (Program Counter - **PC**) de 32 bits e o Registrador de Estado (Status Register - SR) de 16 bits.

A figura 3.1 apresenta como o **registrador de estado** (Status Register - SR) é formado e o significado de cada um de seus bits. Este registrador pode ser acessado como um elemento de 16 bits ou como um elemento de 8 bits (os menos significativos), quando então é denominado de **registrador de condições** (Condition Code Register - CCR).

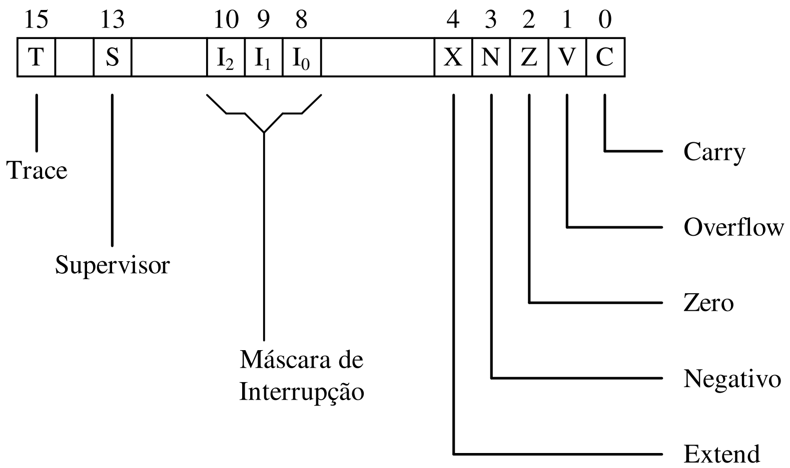

Fig. 3.1: Registrador de Estado.

4.3 Modos de Operação

Uma particularidade do MC68000 refere-se ao dois modos distintos de em que ele pode operar: o modo usuário e o modo supervisor.

O **modo supervisor** é um modo privilegiado de processamento, reservado tipicamente para programas de gerenciamento (sistemas operacionais, monitores, etc.), onde se faz necessário o acesso irrestrito a todos os recursos da máquina.

O **modo usuário** é um modo de processamento não privilegiado, onde certas instruções não são permitidas, bem como determinados tipos de acesso à memória e controle de interrupções.

O processador indica através de três sinais do barramento de controle o seu modo de operação. Assim, é possível restringir, por hardware, o acesso a determinadas posições de memória e/ou interfaces de entrada/saída, que, por serem vitais para a operação do sistema, não devem ser inadvertidamente ou propositalmente alteradas por usuários menos experientes ou mesmo mal-intencionados.

Cada um destes modos de operação possui o seu próprio **registrador apontador de pilha** (A7), como indicado na figura 2.1. Além disto, no modo supervisor, tem-se o acesso a todos os 16 bits do Registrador de Estado (byte do systema e byte do usuário). Já no modo usuário, o acesso a este registrador é restrito ao byte menos significativo (CCR). Note que o bit 13 do registrador de estado estabelece o modo de operação do processador. A alteração deste bit, portanto, só é possível no modo supervisor.

4.4 Organização da Memória

A memória de um micro-computador baseado no MC68000 constitui-se em um arranjo linear de células de memória de 8 bits de comprimento (byte) (fig. 4.1.a). O MC68000 possui capacidade para acessar estas células individualmente (acesso a byte), ou ainda duas células simultaneamente (acesso a word). <u>O acesso a uma palavra só é possível em um endereço par</u>, sendo que o byte mais significativo ocupa este endereço par e o byte menos significativo a posição seguinte (endereço par + 1) (fig. 4.1.b). Um acesso a long word é efetuado através de dois acessos consecutivos a word; sendo portanto definido apenas para endereços pares. No acesso a long word, assim como no acesso a word, a informação mais significativa ocupa a posição de menor endereço (fig. 4.1.c).

a) Bytes

Endereço			
000000	msb	Byte 000000	lsb
000001	msb	Byte 000001	lsb
000002	msb	Byte 000002	lsb
⋮			
FFFFFE	msb	Byte FFFFFE	lsb
FFFFFF	msb	Byte FFFFFE	lsb

b) Words

Endereço					
000000	msb	Byte 000000	Byte 000001	lsb	Word 000000
000002	msb	Byte 000000	Byte 000001	lsb	Word 000001
⋮					
FFFFFE	msb	Byte 000000	Byte 000001	lsb	Word FFFFFE

c) Long Words

Endereço				
000000	msb	Word 000000 - msW		Long Word 000000
		Word 000001 - lsW	lsb	
000004	msb	Word 000002 - msW		Long Word 000001
		Word 000003 - lsW	lsb	
⋮				
FFFFFE	msb	Word 7FFFFE - msW		Long Word 3FFFFF
		Word 7FFFFF - lsW	lsb	

msb - bit mais significativo
lsb - bit menos significativo
msB - byte mais significativo
lsB - byte menos significativo
msW - word mais significativa
lsW - word menos significativa

Fig. 4.1: Organização da memória

4.5 Jogo de Instruções

As instruções de um processador podem ser representadas por dois formatos distintos: o formato binário e o formato mnemônico.

O **formato binário** especifica a configuração de bits que define a instrução. Este formato representa diretamente como a instrução existe dentro do computador. Por outro lado, a programação diretamente neste formato se apresenta como uma tarefa árdua que exige extrema concentração do programador, mesmo levando em conta que o formato binário pode ser condensado através de outras representações numéricas, como por exemplo, a representação octal ou hexadecimal. Neste curso, será utilizada preferencialmente a codificação hexadecimal para a representação do formato binário. Por convenção, os números hexadecimal começaram sempre como o símbolo $.

O **formato mnemônico**, como o próprio nome sugere, representa as instruções de uma forma mnemônica, facilitando a leitura e escrita de programas. O MC68000 apresenta as seguinte operações no formato mnemônico:

MNEMÒNICO	SIGNIFICADO
ABCD	adição BCD (binary coded decima)
ADD	adição binária de dados
ADDA	adição binária de endereços
ADDI	adicão binária com operando imediato
ADDQ	adição binária rápida
ADDX	adição binária com bit de extensão
AND	função lógica E
ANDI	função lógica E com operando imediato
ASL	deslocamento aritmético para a esquerda
ASR	deslocamento shift aritmético para a direita
Bcc	salto condicional
BCHG	testa bit e nega
BCLR	testa bit e zera
BRA	salto incondicional
BSET	testa bit e faz verdadeiro
BSR	salto para subrotina
BTST	testa bit
CHK	testa registrador contra limites
CLR	zera operando
CMP	compara operando com registrador de dados
CMPA	compara operando com registrador de endereço
CMPI	compara operando imediato
CMPM	comparação múltipla
DBcc	testa condição, decrementa e salta
DIVS	divisão com sinal
DIVU	divisão sem sinal
EOR	função lógica OU EXCLUSIVO
EORI	OU EXCLUSIVO com operando imediato
EXG	troca conteúdo entre registradores
EXT	extensão de sinal
ILLEGAL	instrução ilegal
JMP	salto
JSR	salto para subrotina
LEA	carrega endereço efetivo
LINK	"link"
LSL	deslocamento lógico para a esquerda
LSR	deslocamento lógico para a direita
MOVE	transfere dado da fonte para o destino
MOVEA	transfere endereço da fonte para registrador de endereço
MOVEM	transfere conteúdo de conjunto de registradores
MOVEP	transferência de dados para periférico
MULS	multiplicação com sinal
MULU	multiplicação sem sinal
NBCD	nega BCD
NEG	nega
NEGX	nega com bit de extensão
NOP	nenhuma operação
NOT	complemento lógico
OR	função lógica OU
ORI	OU com operando imediato
PEA	empilha endereço efetivo
RESET	inicializa dispositivos periféricos
ROL	roda para a esquerda
ROR	roda para a direita
ROXL	roda para a esquerda com extensão
ROXR	roda para a direita com extensão
RTE	retorno de exceção

RTR	retorno e reposição do registrador de condições
RTS	retorno de subrotina
SBCD	subtração BCD
Scc	faz verdadeiro dependente da condição
STOP	pára
SUB	subtração binária de dado
SUBA	subtração binária de endereço
SUBI	subtração binária com operando imediato
SUBQ	subtração binária rápida
SUBX	subtração binária com bit de extensão
SWAP	troca as palavras (16 bits) de registrador
TAS	testa e faz verdadeiro
TRAP	"trap"
TRAPV	"trap on overflow"
TST	testa
UNLK	"unlink"

Adicionalmente ao mnemônico da operação, faz-se ainda necessário especificar eventuais operandos e seus tamanhos (byte, word, ou long word). A especificacão dos operandos está intimamente relacionada aos modos de endereçamentos suportados pelo processador (veja seção 2.5.2). O tamanho do operando é especificado através das extensões .B (byte) .W (word, em geral o valor default) e .L (long word) anexada ao final do mnemônico da operação. Assim, exemplos de instruções no formato mnemônico seriam:

MOVE.B	D0,D1	copia 8 bits menos significativo (byte) de D0 paraD1.
ADD.W	D2,D3	soma word de D2 com word de D3 e deixa resultado em D3.
NEG.L	D4	nega conteúdo de D4, afetando os 32 bits do registrador.

Para maiores detalhes do jogo de intruções consulte o anexo A.

4.5.1 Tamanho das instruções no formato binário

O tamanho das instruções do MC68000 pode variar de 1 palavra (word) até 5 palavras (fig. 5.1). O tamanho da instrução, assim como a especificação da operação a ser efetuada, é definido sempre pela primeira palavra da instrução. Tal palavra é denominada apropriadamente de palavra de operação (Operation Word). As palavras subsequentes, se existirem, especificam eventuais operandos. O tamanho final de uma instrução depende diretamente do(s) modo(s) de endereçamento utilizado para especificar o(s) operando(s).

15   0
Palavra de Operação (1ª palavra especifica Operação e Modo de Endereçamento)
Operando Imediato (caso exista: 1 ou 2 palavras adicionais)
Extensão do Endereço Efetivo - Operando Fonte (caso exista: 1 ou 2 palavras adicionais)
Extensão do Endereço Efetivo - Operando Destino (caso exista: 1 ou 2 palavras adicionais)

Fig. 5.1: Formato de uma instrução

4.5.2 Modos de Endereçamento

A seguir é apresentada uma descrição dos modos de endereçamento suportado pelo MC68000. Nesta apresentação as seguintes convenções foram adotadas:

An - registrador de endereço (n especifica o número do registrador)
Dn - registrador de dados (n especifica o número do registrador)
Rn - qualquer registrador (dados ou endereço)

PC	- registrador contador de programa
SR	- registrador de estado
CCR	- registrador de condições
SP	- apontador de pilha ativo (pode ser o do supervisor ou o do usuário)
USP	- apontador de pilha do usuário
SSP	- apontador de pilha do supervisor
d8	- valor de deslocamento de 8 bits
d16	- valor de deslocamento de 16 bits
xxx	- valor numérico
xxx.B	- valor numérico representável em 8 bits
xxx.W	- valor numérico representável em 16 bits
xxx.L	- valor numérico representável em 32 bits

4.5.2.1 Modos de Endereçamento Direto a Registador

Nestes modos de endereçamento o operando é um dos 8 registradores de dados ou um dos 8 registradores de endereço.

4.5.2.1.1 Modo de endereçamento direto a registrador de dados

Neste modo de endereçamento é especificado um registrador de dados. O operando é o conteúdo deste registrador de dados.

Formato mnemônico: Dn
Exemplo: MOVE D0,D1

4.5.2.1.2 Modo de endereçamento direto a registrador de endereço

Neste modo de endereçamento é especificado um registrador de endereço. O operando é o conteúdo deste registrador.

Formato menmônico: An
Exemplo: MOVEA A0,A1

4.5.2.2 Modos de Endereçamento à Memória

Este modo de endereçamento permite a especificação de operandos em memória, através da especificação de endereços de memória.

4.5.2.2.1 Modo de endereçamento indireto, via registrador de endereço

O endereço do operando é dado pelo conteúdo de um registrador de endereço.

Formato mnemônico: (An)
Exemplo: MOVE (A0),(A1)

4.5.2.2.2 Modo de endereçamento indireto, via registrador de endereço com pré-decremento

Como no modo de endereçamento anterior, o endereço operando é dado pelo conteúdo de um registrador de endereço. Neste modo de endereçamento, entretanto, o conteúdo do registrador de endereço antes de ser usado é decrementado de 1, 2, ou 4, dependendo se o tamanho do operando é byte, word ou long word.

Formato mnemônico: -(An)
Exemplo: MOVE -(A2),-(A3)

4.5.2.2.3 Modo de endereçamento indireto, via registrador de endereço com pós-incremento

Neste modo de endereçamento, o endereço do operando é fornecido pelo conteúdo do registrador de endereço. Após o acesso ao operando, o conteúdo do registrador de endereço é automaticamente incrementado de 1, 2, ou 4, dependendo se o tamanho do operando era byte, word, ou long word.

Formato mnemônico: (An)+
Exemplo: MOVE (A2)+,(A5)+

4.5.2.2.4 Modo de endereçamento indireto, via registrador de endereço com deslocamento

O endereço do operando é dado pela soma do conteúdo do registador de endereço com o deslocamento especificado. O deslocamento de 16 bits é estendido até 32 bits , conservando-se o sinal (sign-extended). Este modo de endereçamento exige uma palavra adicional para a especificação do deslocamento (veja fig. 6.1).

Formato mnemônico: d16(An)
Exemplo: MOVE 8(A3),$200(A1)

Fig. 6.1: Palavra adicional do modo de endereçamento indireto com deslocamento

4.5.2.2.5 Modo de endereçamento indireto, via registrador de endereço com índice

O endereço do operando é dado pela soma do conteúdo do registrador de endereço com o registrador índice e o deslocamento de 8 bits. O registrador índice pode ser qualquer um dos registradores de dados ou de endereço. O registrador índice pode contribuir com 16 (word) ou 32 bits (long word) para a formação do endereço final. Caso seja especificado um registrador índice de tamanho word, este é estendido até 32 bits, conservando-se o sinal (sign-extended). Da mesma forma o deslocamento de 8 bits é estendido até 32 bits. Este modo de endereçamento exige uma palavra adicional para especificacação do registrador índice e o deslocamento de 8 bits (veja fig. 7.1).

Formato mnemônico: d8(An,Rn.W), ou d8(An,Rn.L)
Exemplo: MOVE $10(A0,D1.W),6(A4,A2.L)

15	14 12	11	10	9	8	7 0
D/A	registrador	W/L	0	0	0	deslocamento

Bit 15 - Indicador de registrador índice
 0 - Registrador de dados
 1 - Registrador de endereço
Bit 14 até 12 - Número de registrador índice
Bit 11 - Tamanho do índice
 0 - 16 bits (lsW) - extensão conservando o sinal
 1 - 32 bits

Fig. 7.1: Palavra adicional do modo de endereçamento indireto com índice

4.5.2.2.6 Modo de endereçamento absoluto curto

Neste modo de endereçamento o endereço do operando é especificado utilizando-se valor de 16 bits. Este endereço é estendido, conservando-se o sinal (signl-extended), antes de ser usado. Este modo de endereçamento exige uma palavra adicional para a especificação dos 16 bits do endereço absoluto curto.

Formato mnemônico: xxx.W
Exemplo: MOVE $300,20

15 0
endereço absoluto curto

Fig. 8.1: Palavra adicional modo de endereçamento absoluto curto

4.5.2.2.7 Modo de endereçamento absoluto longo

Neste modo de endereçamento o endereço é especificado usando-se 32 *bits*. Este modo de endereçamento exige duas palavras adicionais para a especificação dos 32 *bits* do endereço absoluto longo.

Formato mnemônico: xxx.L
Exemplo: MOVE 70000,$10000

31 16	endereço absoluto
15 0	longo

Fig. 9.1: Palavra adicional modo de endereçamento absoluto longo

4.5.2.2.8 Modo de endereçamento indireto, via PC com deslocamento (relativo ao PC)

O endereço do operando é dado pela soma do conteúdo do PC com o deslocamento de 16 bits. O valor do deslocamento é estendido conservando-se o sinal antes de ser usado. Este modo de endereçamento exige uma palavra adicional para especificação dos 16 bits do deslocamento.

Formato mnemônico: d16(PC)
Exemplo: JMP $100(PC)

Cuidado: Este modo de endereçamento merece um comentário no seu uso. O valor do deslocamento que é armazenado no código de instrução pode não ser o valor que é escrito na instrução no formato mnemônico. Isto é dependente do tradutor. No caso do programa Tutor usado neste curso (veja seção 5), o valor escrito na instrução não é o deslocamento, mas sim o valor efetivo do endereço. O deslocamento codificado na instrução é calculado subtraindo o valor do PC+2 do valor efetivo. Por exemplo, vamos supor que o valor do deslocamento seja de 10 bytes. Assim a instrução será codificada como $4EFA:000A sendo a segunda palavra o valor do deslocamento. Se esta instrução reside no endereço $1000 por exemplo, qual é o endereço efetivo que a UCP será desviada? Será dada pelo $1000 mais 2 mais 10, que é igual a $1000+$0002+$000A = $100C. Então no formato mnemônico a instrução é escrita como JMP $100C(PC). É importante perceber que o valor $100C não é o deslocamento mas sim o endereço da próxima instrução a ser executada. Outro aspecto importante é que instruções com este tipo de endereçamento relativo ao PC são instruções ditas relocáveis pois caso esta instrução $4EFA 000A seja reposicionada para o endereço $2000 por exemplo, o desvio será feito para o endereço $200C sem qualquer mudança na codificação da instrução.

4.5.2.2.9 Modo de endereçamento indireto, via PC com índice (relativo ao PC com índice)

O endereço do operando é dado pela soma do PC com registrador índice e deslocamento de 8 bits. O registrador índice pode ser qualquer um dos registradores de dados ou endereço. O registrador índice pode contribuir com 16 (word), ou 32 bits (long word), para a formação do endereço final. Caso seja usado um registrador índice de tamanho word, este é estendido conservando-se o sinal. Da mesma forma o deslocamento de 8 bits é estendido até 32 bits. Este modo de endereçamento exige uma palavra adicional para especificação do registrador índice e os 8 bits do deslocamento (veja fig. 7.1).

Formato menmônico: d8(PC,Rn.W), ou d8(PC,Rn.L)
Exemplo: JMP $1040(PC,A0.L)

Cuidado: Como explicado no modo indireto via PC, neste modo o valor escrito na instrução usando o programa Tutor já é o endereço levando-se em conta o valor do PC. No exemplo acima, supondo que a instrução esteja no endereço $1000 e o registrador A0.L tenha valor 2, o deslocamento a ser codificado na instrução é $1040-($1000+2). O endereço efetivo é dado por $1040 mais o valor de A0, i.e., $1042.

4.5.2.3 Modos de Endereçamento Especiais

4.5.2.3.1 Modo de endereçamento imediato

Este modo de endereçamento permite a especificação do valor do operando diretamente. Pode-se ter operandos imediatos de tamanho byte, word ou long word. Este modo de endereçamento exige 1 palavra adicional se o tamanho do operando for byte ou word, e 2 palavras adicionais se for do tamanho long word.

Formato mnemônico: #xxx
Exemplo: MOVE #$100,D0

4.5.2.3.2 Referência implícita

Algumas instruções do MC68000 permite a referência implicita ao PC, SP, SSP, USP, ou SR.

Exemplo: BRA $100

4.5.3 Exemplos de Instruções

A seguir são apresentados exemplos de instruções tanto no formato mnemônico quanto no binário (hexadecimal). O leitor é encorajado a checar a correção da codificação hexadecimal.

4.5.3.1 Instruções de tamanho 1

a) Instrução sem operando. Exemplos:

mnemônico		hexadecimal
NOP		$4E71
RTS		$4E75

b) Instrução com único operando, especificado por modo de endereçamento direto, via registrador de dados. Exemplos:

mnemônico		hexadecimal
EXT	D0	$4880
CLR.B	D0	$4200

c) Instrução com único operando, especificado por modo de endereçamento direto, via registrador de endereço. Exemplo:

mnemônico		hexadecimal
UNLK	A0	$4E58

c) Instrução com único operando, especificado por modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento). Exemplos:

mnemônico		hexadecimal
NOT.B	(A0)	$4610
LSL	(A0)+	$E3D8

d) Instrução com um operando, especificado por modo de endereçamento imediato. Exemplos:

mnemônico		hexadecimal
TRAP	#14	$4E4E

d) Instrução com dois operandos, ambos especificados por modo de endereçamento direto. Exemplos:

mnemônico		hexadecimal
ABCD	D0,D1	$C300
ADDA.L	D0,A1	$D3C0

e) Instrução com dois operandos, um especificado por modo de endereçamento direto, e o outro, por modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento). Exemplos:

mnemônico		hexadecimal
AND.B	D0,-(A1)	$C121

SUB.W	(A0)+,D1	$9258

f) Instrução com dois operandos, ambos especificados por modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento). Exemplos:

mnemônico	hexadecimal	
ABCD.B	-(A0),-(A1)	$C308
CMPM.L	(A0)+,(A1)+	$B388

g) Instrução com dois operandos, um especificado por modo de endereçamento imediato (quick), e o outro, por modo de endereçamento direto, ou indireto (com ou sem pré-decremento/pós-incremento). Exemplos:

mnemônico	hexadecimal	
SUBQ.L	#3,(A0)+	$5798
MOVEQ.L	#-1,D0	$70FF

4.5.3.2 Instruções de tamanho 2

a) Instrução com um operando, especificado por modo de endereçamento indireto via registrador de endereço com deslocamento. Exemplos:

mnemônico	hexadecimal	
ASL	-1(A0)	$E1E8 FFFF
NEG.L	15(A0)	$44A8 000F

b) Instrução com um operando, especificado por modo de endereçamento indireto, via registrador de endereço com índice. Exemplos:

mnemônico	hexadecimal	
NBCD	-2(A0,D1.W)	$4830 10FE
ROL.W	-23(A0,A1.L)	$E7F0 98E9

c) Instrução com um operando, especificado por modo de endereçamento absoluto curto. Exemplos:

mnemônico	hexadecimal	
NOT.W	$1	$4678 0001
SNE	$7FFF.	$56F8 7FFF

d) Instrução com um operando, especificado por modo de endereçamento indireto, via PC com deslocamento. Exemplos (Obs.: as instruções residem no endereço $1000):

mnemônico	hexadecimal	
JMP	$900(PC)	$4EFA F8FE
PEA	16(PC)	$487A F00E

Obs: veja os cuidados necessários a serem observados neste modo de endereçamento relativo ao PC como discutido no ítem 4.5.2.2.8.

e) Instrução com um operando, especificado por modo de endereçamento indireto, via PC com índice. Exemplos (Obs .: as instruções residem no endereço $1000):

mnemônico	hexadecimal	
JSR	$1010(PC,DO.W)	$4EBB 000E
JMP	$FF0(PC,A0.L)	$4EFB 88EE

f) Instrução com um operando, especificado por modo de endereçamento imediato - tamanho byte ou word. Exemplo:

mnemônico	hexadecimal	
STOP	#2000	$4E72 07D0

g) Instrução com dois operandos, um especificado por modo de endereçamento direto ou modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento), e o outro, por modo de endereçamento indireto via registrador de endereço com deslocamento. Exemplos:

mnemônico	hexadecimal	
ADD	D0,$3000(A1)	$D169 3000
MULS	$20(A0),D1	$C3E8 0020

h) Instrução com dois operandos, um especificado por modo de endereçamento direto ou modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento), e o outro, por modo de endereçamento indireto, via registrador de endereço com índice. Exemplos:

mnemônico	hexadecimal	
MOVE.B	(A0)+,-1(A1,D2)	$1398 20FF
SUB.L	D0,$30(A2,A1)	$91B2 9030

i) Instrução com dois operandos, um especificado por modo de endereçamento direto ou modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento), e o outro, por modo de endereçamento absoluto curto. Exemplos:

mnemônico	hexadecimal	
BTST	D0,$2000	$0138 2000
OR.B	1024,D0	$8038 4000

j) Instrução com dois operandos: a) um especificado por modo de endereçamento direto ou modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento), b) o outro por modo de endereçamento indireto, via PC com deslocamento. Exemplos (Obs.: as instruções residem no endereço $1000):

mnemônico	hexadecimal	
MOVE.B	$1000(PC),D0	$103A FFFE
ADD.L	$0(PC),D0	$D0BA EFFE

k) Instrução com dois operandos: a) um especificado por modo de endereçamento direto ou modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento); b) o outro por modo de endereçamento indireto, via PC com índice. Exemplos (Obs.: as instruções residem no endereço $1000):

mnemônico	hexadecimal	
DIVU	$1030(PC,A1),DO	$80FB 902E

OR.W	$1030(PC,A1),D0	$807B 902E

l) Instrução com dois operandos: a) um especificado por modo de endereçamento direto ou modo de endereçamento indireto, via registrador de endereço (com ou sem pré-decremento/pós-incremento); b) o outro por modo de endereçamento imediato - tamanho byte ou word. Exemplos:

mnemônico		hexadecimal
CMPA	#$FFFF,A0	$B0FC FFFF
SUBI.W	#$10,-(A0)	$0460 0010

4.5.3.3 Instruções de tamanho 3

a) Instrução com um operando especificado por modo de endereçamento absoluto longo. Exemplos:

mnemônico		hexadecimal
JMP	$FFFFFE	$4EF9 00FF FFFE
NOT.B	$10000	$4639 0001 0000

b) Instrução com dois operandos: a) um especificado por modo de endereçamento direto, ou modo de endereçamento indireto, via registrador de endereço; (com ou sem pré-decremento/pós-incremento), b) o outro por modo de endereçamento absoluto longo. Exemplos:

mnemônico		hexadecimal
BCLR	D0,$30000	$01B9 0003 0000
AND.L	$FFFFFF,D0	$C0B9 00FF FFFF

c) Instrução com dois operandos, ambos especificados por modo de endereçamento absoluto curto. Exemplo:

mnemônico		hexadecimal
MOVE	$3000,$3500	$31F8 3000 3500

d) Instrução com dois operandos, ambos especificados por modo de endereçamento indireto, via registrador de endereço com deslocamento. Exemplo:

mnemônico		hexadecimal
MOVE.L	$20(A0),-1(A1)	$2368 0020 FFFF

e) Instrução com dois operandos, ambos especificados por modo de endereçamento indireto, via registrador de endereço com índice. Exemplo:

mnemônico		hexadecimal
MOVE.B	-1(A0,D1),1(A2,D3.L)	$15B0 10FF 3801

f) Instrução com dois operandos: a) um especificado por modo de endereçamento absoluto curto, b) o outro por modo de endereçamento indireto, via registrador de endereço com deslocamento. Exemplo:

mnemônico hexadecimal

MOVE.B $4000,20(A0) $1178 4000 0014

g) Instrução com dois operandos: a) um especificado por modo de endereçamento absoluto curto, b) o outro por modo de endereçamento indireto, via registrador de endereço com índice Exemplo:

mnemônico		hexadecimal
MOVE.B	-2(A0,D1.L),$20	$11F0 18FE 0020

h) Instrução com dois operandos: a) um especificado por modo de endereçamento absoluto curto, b) o outro por modo de endereçamento indireto, via registrador PC com deslocamento. Exemplo (Obs.: a instrução reside no endereço $1000):

mnemônico		hexadecimal
MOVE	$38(PC),$100	$33FA F036 0100

i) Instrução com dois operandos: a) um especificado por modo de endereçamento absoluto curto, b) o outro por modo de endereçamento indireto, via registrador PC com índice. Exemplo(Obs.: a instrução reside no endereço $1000):

mnemônico		hexadecimal
MOVE.L	$FF0(PC,A2),$58	$21FB A0EE 0058

j) Instrução com dois operandos: a) um especificado por modo de endereçamento indireto, via registrador de endereço com deslocamento, b) o outro por modo de endereçamento indireto, via registrador de endereço com índice. Exemplo:

mnemônico		hexadecimal
MOVE	$20(A0,D5.L),-2(A3)	$3770 5820 FFFE

k) Instrução com dois operandos: a) um especificado por modo de endereçamento indireto, via registrador de endereço com deslocamento, b) o outro por modo de endereçamento indireto, via registrador PC com deslocamento. Exemplo (Obs.: a instrução reside no endereço $1000):

mnemônico		hexadecimal
MOVE.L	-8(PC),$7F00(A2)	$257A EFF6 7F00

l) Instrução com dois operandos: a) um especificado por modo de endereçamento indireto, via registrador de endereço com deslocamento, b) o outro por modo de endereçamento indireto, via registrador PC com índice. Exemplo (Obs.: a instrução reside no endereço $1000):

mnemônico		hexadecimal
MOVE	$1081(PC,D4),18(A6)	$3D7B 407F 0012

m) Instrução com dois operandos: a) um especificado por modo de endereçamento indireto, via registrador de endereço com índice, b) o outro por modo de endereçamento indireto, via registrador PC com deslocamento. Exemplo(Obs.: a instrução reside no endereço $1000):

mnemônico		hexadecimal
MOVE	10(PC),$7F(A4,D2)	$39BA F008 207F

n) Instrução com dois operandos: a) um especificado por modo de endereçamento indireto, via registrador de endereço com índice, b) o outro por modo de endereçamento indireto, via registrador PC com índice. Exemplo (Obs.: a instrução reside no endereço $1000):

mnemônico		hexadecimal
MOVE.L	$1002(PC,D0),(A0,D0)	$21BB 0000 0000

o) Instrução com dois operandos: a:) um especificado por modo de endereçamento imediato - tamanho byte ou word, b) o outro por modo de endereçamento absoluto curto. Exemplo:

mnemônico		hexadecimal
MOVE.W	#2000,$1000	$31FC 07D0 1000

p) Instrução com dois operandos: a) um especificado por modo de endereçamento imediato - tamanho byte ou word, b) o outro por modo de endereçamento indireto, via registrador de endereço com deslocamento ou com índice. Exemplo:

mnemônico		hexadecimal
MOVE.B	#$70,(A0,D1)	$11BC 0070 1000

4.5.3.4 Instruções de tamanho 4

a) Instrução com dois operandos: a) um especificado por modo de endereçamento absoluto longo, b) o outro por modo de endereçamento indireto, via registrador de endereço com deslocamento. Exemplo:

mnemônico		hexadecimal
MOVE.B	$123456,16(D0)	$1179 0012 3456 0010

b) Instrução com dois operandos: a) um especificado por modo de endereçamento absoluto longo, b) o outro por modo de endereçamento indireto, via registrador de endereço com índice. Exemplo:

mnemônico		hexadecimal
MOVE	2(A1,D0.L),$00102000	$33F1 0802 0010 2000

c) Instrução com dois operandos: a) um especificado por modo de endereçamento absoluto longo, b) o outro por modo de endereçamento indireto, via registrador PC com deslocamento. Exemplo (Obs.: a instrução reside no endereço $1000):

mnemônico		hexadecimal
MOVE.B	$1101(PC),$10000	$13FA 00FF 0001 0000

d) Instrução com dois operandos: a) um especificado por modo de endereçamento absoluto longo, b) o outro por modo de endereçamento indireto, via registrador PC com índice. Exemplo(Obs.: a instrução reside no endereço $1000):

mnemônico		hexadecimal
MOVE.L	$1004(PC,D0.L),$20000	$13FB 0802 0002 0000

e) Instrução com dois operandos: a) um especificado por modo de endereçamento imediato - tamanho longo, b) o outro por modo de endereçamento indireto, via registrador de endereço com deslocamento. Exemplo:

mnemônico		hexadecimal
MOVE.L	#$12345678,16(A0)	$217C 1234 5678 0010

f) Instrução com dois operandos: a) um especificado por modo de endereçamento imediato - tamanho longo, b) o outro por modo de endereçamento indireto, via registrador de endereço com índice. Exemplo:

mnemônico		hexadecimal
MOVE.L	#0,2(A1,D0.W)	$23BC 0000 0000 0002

4.5.3.5 Instruções de tamanho 5

a) Instrução com dois operandos, ambos especificados por modo de endereçamento absoluto longo. Exemplo:

mnemônico		hexadecimal
MOVE.B	$10000,$FFFFFF	$13F9 0001 0000 00FF FFFF

a) Instrução com dois operandos: a) um especificado por modo de endereçamento imediato - tamanho longo, b) o outro por modo de endereçamento absoluto longo. Exemplo:

mnemônico		hexadecimal
MOVE.L	#$87654321,$123456	$23FC 8765 4321 0012 3456

5. O Firmware TUTOR

O sistema SBC68K posssui ainda firmware de auxílio ao desenvolvimento de programas. Tal firmware, denominado TUTOR, permite ao usuário inserir, monitorar e depurar os seus programas.

O TUTOR aceita linhas de comando no seguinte formato (observar que uma linha deve ser sempre terminada com um ENTER):

TUTOR 1.32> [NO] <comando> [<parâmetros>] [;<opções>]

onde:

TUTOR 1.32>	é o prompt gerado pelo TUTOR, indicando que está apto a receber um comando
NO	prefixo de negação do comando que se segue
comando	é o comando (veja lista de comandos abaixo)
parâmetros	são separados por espaços e podem representar uma expressão aritmética ou um endereço
opções	dependendo do comando, múltiplas opções podem ser selecionadas

Os comandos do TUTOR podem ser classificados nas seguintes categorias:

- Comandos que permitem ao usuário visualizar e modificar posições de memória.
- Comandos que permitem ao usuário visualizar e modificar os registradores internos do MC68000.
- Comandos que permitem ao usuário executar o programa sob vários níveis de controle.
- Comandos que controlam acessos a vários dispositivos de entrada/saída ligados a placa.

Os comandos do TUTOR que serão mais utilizados neste curso são apresentados na tabela que se segue:

Comando	Descrição	Página do Manual
MD	Memory Display: mostra o connteúdo de posições de memória	2-23
MM	Memory Modify: permite visualizar e modificar posições de memória	2-24
MS	Memory Set: permite modificar posições de memória	2-27
DF	Display Formatted Registers: permite visualizar o conteúdo de todos os registradores internos da UCP.	2-16
.A0 - .A7	pemite visualizar e modificar o conteúdo dos registradores de endereços	2-35
.D0 - .D7	pemite visualizar e modificar o conteúdo dos registradores de dados	2-35
.PC	permite visualizar e modificar o conteúudo do registrador PC (program counter)	2-35
.SR	permite visualizar e modificar o conteúdo do registrador SR (status register - registrador de estado)	2-35
.SS	permite visualizar e modificar o conteúdo do registrador apontador de pilha do supervisor (SSP - supervisor stack pointer)	2-35
.US	pemite visualizar e modificar o conteúdo do registrador apontador de pilha do usuário (USP - user stack pointer)	2-35
BF	Block of Memory Fill: permite preencher um bloco de memória com um valor constante.	2-10
BM	Block of Memory Move: permite copiar o conteúdo de um bloco de memória.	2-11
DC	Data Conversion: permite a conversão de expressões aritméticas para valores decimais e hexadecimais.	2-15
BR	Breakpoint Set: permite a definição de breakpoints (pontos de parada).	2-12
GD	Go Direct: permite a execução de um programa, sem utilizar breakpoints.	2-18
GO, G	Go: permite a execução de um programa, parando em eventuais breakpoints definidos previamente pelo usuário.	2-19
GT	Go unTil breakpoint: permite a execução de um programa com a definição de um breakpoint.	2-20
NOBR	permite retirar breakpoints.	2-28
TR, T	Trace: permite a execucão passo-a-passo de um programa.	2-38
TT	Trace To breakpoint: permite a execução passo-a-passo de um programa até um breakpoint.	2-39
PA	Printer Attach: faz com que toda informação enviada	2-31

	para a tela do terminal seja também impressa em uma impressora.	
NOPA	No Printer Attach: cancela o efeito do comando PA.	2-29

Uma descrição completa dos comando suportados pelo TUTOR é apresentada no capítulo 2 da referência [3].]

6. Comunicação PC - SBC68K

Neste labororatório, micro-computadores da classe PC (Personal Computer) serão utilizados como terminais para os micro-computadores SBC68K. Um terminal é entendido aqui como o conjunto formado pelo teclado (dispositivo de entrada através do qual o usuário pode digitar os comandos do TUTOR) e pelo monitor (dispositivo de saída por onde o usuário receberá informação textual geral pelo TUTOR). Para configurar um PC para atuar com terminal do SBC68K seguir os seguintes passos:

1. Ligar o PC;
2. Ligar o Monitor;
3. Ligar o SBC68K; e
4. No prompt do DOS digitar **terminal**.

Terminal é um programa de comunicação desenvolvido pelo Prof. J. Mario De Martino que, em estando em execução no PC, permite transmissão e recepção de dados através de sua porta serial A (COM1:). Este programa faz com que todo caráter digitado no teclado seja enviado para porta serial A do PC, e que todo caráter recebido nesta esta porta seja apresentado na tela do monitor. Na configuração do laboratório, a porta serial A do PC está (deve estar) conectada à porta serial TERMINAL do SBC68K (o conector desta porta está identificado na placa do SBC68K com a inscrição TERMINAL). Após a execução do programa terminal, o usuário pode através do terminal formado pelo PC interagir diretamente com o programa TUTOR do SBC68K (veja seção 5), abstraindo a existência do PC.

Para terminar o programa **terminal** e retornar ao DOS digitar CTRL-C (apertar simultaneamente as teclas Ctrl e a tecla C).

7. Exercícios

1. (0,5 pt) Calcule os resultados das expressões abaixo. O resultado deve ser expresso em hexadecimal. Considerar representação com 16 bits em complemento de dois:

(a) $2000 + $14
(b) $2000 + 14
(c) $2000 - 4
(d) 4 - $2000
(e) 1 + $FFFF

2. (0, 5 pt) Quais são os valores máximo e mínimo possíveis de serem representados por cada tipo de dado abaixo. Considere dois casos: números sem sinal e números com sinal em complemento de 2. Para cada caso, escreva os valores em hexadecimal e decimal

(a) Byte (8bits),
(b) Word,(16 bits) e

(c) Long Word (32 bits)

3. (0, 5 pt) O SBC68K está configurado com 64Kbytes de memória RAM. Esta memória reside a partir do endereço $000000. Pergunta-se:

(a) Qual é o endereço final desta memória?
(b) Indique o endereço inicial e final da memória RAM disponível para o usuário? O trecho não disponível para o usuário é de uso reservado do TUTOR(consulte o capítulo 2 da referência /3/).

4. (0, 5 pt) O SBC68K dispõe também de memória do tipo ROM. Nesta memória reside o programa TUTOR. Por que é conveniente que o TUTOR resida em memória ROM ?

5. (0, 5 pt) O MC68000 possui um barramento de endereço de 24 bits. Pergunta:

(a) Qual o tamanho máximo possível, em bytes, de sua memória principal ?
(b) Qual é o endereço inicial e o endereço final desta memória (em hexadecimal) ?
(c) Os registradores de endereço do 68000 possuem 32 bits. Como estes 32 bits são colocados nos 24 bits do barramento de endereço ?

6. (0,5 pt) Explique cada um dos *bits* do código binário das seguintes instruções::

(a) RTS (seção 4.5.3.1 item a)
(b) CLR.B D0 (seção 4.5.3.1 item b)
(c) BTST D0,$2000 (seção 4.5.3.2 item i)
(d) CMPA #$FFFF,A0 (seção 4.5.3.2 item l)
(e) ASL -1(A0) (seção 4.5.3.2 item a)
(f) JMP $900(PC) (seção 4.5.3.2 item d - lembrar que a instrução reside no endereço $1000)
(g) MOVE.B (A0)+,-1(A1,D2) (seção 4.5.3.2 item h)
(h) AND.L $FFFFFF,D0 (seção 4.5.3.3 item b)
(i) MOVE.L #$12345678,16(A0) (seção 4.5.3.4 item e)
(j) MOVE.B $10000,$FFFFFF (seção 4.5.3.5 item a)

7. (0,5 pt) Dar o endereço efetivo dos operandos das instruções apresentadas abaixo.

(a) BTST D0,$2000
(b) ASL -1(A0); suponha A0 = $500F
(c) JMP $900(PC); suponha que a instrução reside no endereço $1000
(d) MOVE.B (A0)+,-1(A1,D2); suponha A0 = $500F, A1 = $501F e D2 = 2
(e) MOVE.L #$12345678,16(A0); suponha A0 = $5000

8. (0,5 pt) Considerando as instruções abaixo:

(a) Identifique os modos de endereçamentos utilizados nas instruções abaixo e explique as diferenças entre estes dois modos de endereçamento.
(b) As duas instruções abaixo carregam o registrador A0 com o mesmo valor. Explique.

```
MOVEA.L  #$200000,A0
LEA      $200000,A0
```

9. (0, 5 pt) No seguinte programa escrito em linguagem de montador (assembly), determine o tamanho em *bytes* de cada instrução e valor do rótulo AQUI (Suponha que o programa esteja armazenado a partir do endereço $4000)

```
        MOVE      #60,D3
        MOVE.L    $3500,A2
        MOVE.B    #00,D2
AQUI:   JSR       $5000
        MOVE.B    D2, (A2)+
        DBF       D3,AQUI
        MOVE.B    #228,D7
        TRAP      #14
```

10. (0, 5 pt) Considerando o trecho de programa abaixo, responda:

a) Quantas vezes a instrução MOVE.W #1234,(A0) será executada?
b) Qual o trecho de memória que será preenchida pelo programa, após este ser executado. Dê o endereço inicial e final.
c) Qual é o conteúdo das posições de memória do trecho preenchido?

```
        MOVEA.L   #$2000,A0
        MOVE.B    #06,D0
LOOP:   MOVE.W    #1234,(A0)
        ADDA      #2,A0
        SUB.B     #1,D0
        CMP.B     #-1,D0
        BNE       LOOP
```

11. (0, 5 pt) Considere agora este outro trecho de programa, contendo um laço (*loop*) de apenas duas instruções. Responda as questões abaixo, levando em conta a comparação deste trecho com o do exercício anterior:

(a) Quantas vezes a instrução MOVE.W #1234,(A0)+ será executada?
(b) Por que não é necessária a instrução que incrementa o registrador de endereço A0?

(c) Por que não é necessária a instrução que decrementa o registrador D0, aqui usado como contador da iteração?
(d) Por que não é necessária a instrução de comparação para testar se o contador chegou a 1?
(e) Por que a inicialização do registrador D0 **NÃO** pode ser efetuada com um MOVE.B ?

```
        LEA       $2000,A0
        MOVE.W    #06,D0
LOOP:   MOVE.W    #1234,(A0)+
        DBF       D0,LOOP
```

Obs: Estes programas de iteração, envolvendo laços (*loops*) são muito comuns e o esquema acima é utilizado com frequência.

12. (0,5 pt) Considere as seis instruções de desvio apresentadas abaixo. Considere, também, que as instruções residem no endereço $1000. Observando que as três primeiras instruções efetuam o desvio para o endereço $2000 e as três últimas para o endereço $1010, responda:

(a) Quais as formas de endereçamento utilizadas em cada instrução ?
(b) Por que a instrução BRA $1010 é codificada apenas em uma palavra ?
(c) Por que a instrução BRA não poderia ser utilizada para efetuar um desvio para a posição $20000 ?
(d) A restrição acima também é válida para a instrução JMP ? Por quê ?
(e) O que é código relocável ?
(f) Os trechos de programa apresentados nas questões 11 e 12 são relocáveis ? Por quê ?
(g) Das seis instruções apresentadas, quais são as relocavéis ?

Hexadecimal	Mnemônico
$4EF8 2000	JMP $2000
4EFA 0FFE	JMP $2000(PC)
$6000 0FFE	BRA $2000
$4EF8 1010	JMP $1010
$4EFA 000E	JMP $1010(PC)
$600E	BRA $1010

13. (0,50 pt) O que faz a seguinte sequência de comandos do TUTOR (Obs.: Conectar a impressora ao SBC68K e ligá-la):

(a) PA
MD 2000 10
NOPA
MD 2000 10
(b) MD3 2000 10

14. (0, 5 pt) Escreva a linha de comando do TUTOR que permita mostrar:

(a) 32 Bytes a partir da posição $2000, e
(b) 32 Words a partir da posição $2000

15. (0, 5 pt) Descreva como inserir a sequência de bytes abaixo a partir do endereço $1000 utilizando o comando MM do TUTOR.

00 01 02 03 04 05 06 07 08 09

16. (0,5 pt) Descreva como mover os dados inseridos no exercício anterior para o endereço $2000 utilizando o comando BM do TUTOR. Os dados originais são destruídos pelo comando BM ?

17. (0, 5 pt) Qual a linha de comando do TUTOR que permite inspecionar o conteúdo de todos os registradores simultaneamente ?

18. (0,5 pt) Descreva como modificar o conteúdo dos registradores para os valores especificados abaixo utilizando os comandos do TUTOR.

A0 = $A5A5A5A5 A1 = 2048 A2 = $400

D0 = 5000 D1 = 1001 D2 = 15000
PC = $2000

19. (0,5pt) Descreva como preencher a região de memória que vai do endereço $3000 até o $30FF (inclusive) com o valor $0A utilizando o comando BF do TUTOR.

20. (0,5 pt) Os comandos MM e MD do TUTOR admitem a opção ;DI. Explique a finalidade e a forma de uso desta opção.

8. Referências

[1] Motorola Inc.: *M68000 8-/16-/32-Bit Microprocessor User's Manual*; 8th Edition, Prentice Hall, 1990.

[2] Wagner Ideali: *68000 Família de Microprocessadores - 32 bits - Software vol. 2*, Livros Érica Editora Ltda, 1987.

[3] Arnewsh Inc.: *SBC68K User's Manual Revision 1.1*, 2nd Edition, 1990.

ANEXO A

JOGO DE INSTRUÇÕES