UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Bruno Henrique Zara
João Victor Millane
Vinicius Augusto Borgue

MOS 6502: Arquitetura e Organização do CPU

São José do Rio Preto
2023

Bruno Henrique Zara
João Victor Millane
Vinicius Augusto Borgue

MOS 6502: Arquitetura e Organização do CPU

Análise dos designs de arquitetura e organização que fizeram o MOS 6502 ser um dos CPUs 8bits mais utilizados da indústria

Orientador: Prof. Dr. Geraldo Francisco Donegá Zafalon

São José do Rio Preto
2023

Sumário

1 Introdução

O *MOS Technology 6502* é um microprocessador de 8 bits que foi projetado por um time pequeno liderado por Charles Ingerham Peddle para a empresa *MOS Technology*. O time de design de arquitetura trabalhou anteriormente na *Motorola*, no projeto *Motorola 6800* que foi o principal processador utilizado em rádios de bolso na época. O *6502* é essencialmente um projeto simplificado, mais barato e mais rápido que o *6800*.

Quando foi lançado em 1975, o *6502* era o microprocessador mais barato do mercado por uma considerável margem. Inicialmente era vendido por menos de um sexto (1/6) do custo de arquiteturas de empresas maiores, como o *Motorola 6800* ou o *Intel 8080*. Seu lançamento causou uma rápida queda de preços em todo o mercado de processadores. Juntamente com o *Zilog Z80*, primeiro processador da *startup Zilog*, provocou uma serie de projetos que resultaram na revolução dos computadores pessoais no inicio da década de 1980.

Os videogames e computadores na década de 1980 e começo da década de 1990, como o *Atari 2600*, familia *Atari 8bit*, *Apple I* e *II*, *Nintendo Entertainment System (NES)*, *Commodore 64*, entre outros, utilizam o *6502* ou variações da arquitetura básica. Logo após o lançamento do *6502*, a empresa *MOS Technology* foi comprada pela *Commodore International*, que continuou a venda do microprocessador e licenças pra outros fabricantes.

Em 1981, a *Western Design Center* começou o desenvolvimento de uma versão feita em *CMOS*, o *65C02*. Esse continua a ser altamente utilizado em sistemas embarcados, com produção estimada na casa dos 100 milhões de unidades.

Portanto, nesse trabalho, será discutida a importância do processador *MOS Technology 6502* e como seu design e eficiência afetaram para sempre a indústria de tecnologia e desenvolvimento de software.

2 História

O *6502* foi projetado por muitos dos mesmos engenheiros que haviam projetado a família de microprocessadores *Motorola 6800*. A *Motorola* iniciou o projeto do microprocessador *6800* em 1971, com Tom Bennett como o principal arquiteto. O projeto do chip começou no final de 1972, os primeiros chips *6800* foram fabricados em fevereiro de 1974 e a família completa foi oficialmente lançada em novembro de 1974. John Buchanan foi o designer do chip *6800*, e Rod Orgill, que mais tarde fez o *6501*, ajudou Buchanan com análises de circuitos e layout do chip. Bill Mensch ingressou na *Motorola* em junho de 1971 e teve um papel importante na definição dos circuitos integrados periféricos para a família 6800.

A *Motorola* visava clientes estabelecidos na indústria eletrônica, como *Hewlett-Packard*, *Tektronix*, *TRW* e *Chrysler*. Eles compartilharam detalhes do sistema de microprocessador de 8 bits proposto com *ROM*, *RAM* e interfaces paralelas e seriais com seus clientes em 1972. A estratégia da *Motorola* não focava no preço do microprocessador, mas sim em reduzir o custo total de design do cliente, oferecendo suporte de desenvolvimento e treinamento. A empresa enfrentou uma competição inicial com a *Intel*, mas ambos reduziram seus preços anunciados para microprocessadores.

Chuck Peddle, um membro da equipe da *Motorola*, percebeu que os clientes estavam desanimados com os altos custos dos chips de microprocessador e começou a esboçar o design de um microprocessador de baixo custo e tamanho reduzido. No entanto, a gerência da Divisão de Produtos Semicondutores da *Motorola* não demonstrou interesse em sua proposta.

Peddle eventualmente se afastou da *Motorola* e se juntou à *MOS Technology*, uma nova empresa fundada por ex-colegas da *Motorola*. Eles desenvolveram o microprocessador *6502*, que era mais acessível e acabou sendo amplamente utilizado em computadores pessoais e consoles de videogame, como o *Commodore 64*, o *Atari 2600* e outros. A *MOS Technology* enfrentou uma ação judicial da *Motorola*, mas acabou resolvendo o caso e continuou a produzir o *6502*.

O *6502* teve um grande impacto na indústria de microprocessadores e foi usado em uma variedade de dispositivos, desde computadores pessoais até consoles de jogos. Sua arquitetura simplificada e custo acessível o tornaram uma escolha popular para muitos fabricantes de eletrônicos.

3 Especificação técnica

3.1 Propriedades

O *6502* é um processador de 8 bits de ordem pequena (*little-endian*) com um barramento de endereço de 16 *bits*. As versões originais foram fabricadas usando tecnologia de processo de 8 µm, com um tamanho de *die* de 3,9 mm x 4,3 mm, totalizando uma área de 16,6 mm^2. Apesar das baixas velocidades de *clock* (tipicamente entre 1 e 2 MHz), o desempenho do *6502* era competitivo com outros *CPUs* contemporâneos que usavam *clocks* significativamente mais rápidos, graças a uma máquina de estados simples implementada por lógica combinacional.

O *6502* tinha registros limitados, incluindo um acumulador de 8 bits, dois registradores de índice de 8 bits, um registrador de *status* do processador com 7 *flags* e um contador de programa de 16 bits. Para compensar a falta de registros, o *6502* incluía um modo de endereçamento de "página zero"que permitia o acesso rápido às primeiras 256 *bytes* de *RAM* com instruções mais curtas.

O espaço de endereço da pilha era fixo na página de memória $01, e instruções especiais eram usadas para empilhar ou desempilhar dados e *status* do processador. O *6502* também tinha vários modos de endereçamento, incluindo modos absolutos, relativos, de acumulador e imediatos.

O processador era capaz de realizar adições e subtrações em binário ou decimal codificado em binário (*BCD*). Em modo *BCD*, as operações de adição resultavam em valores de 8 bits com o *bit* de transporte definido, enquanto em modo binário, o *bit* de transporte era zerado.

As instruções do *6502* eram codificadas em *opcodes* de 8 *bits*, com *bits* específicos definindo a operação e o modo de endereçamento. O *6502* usava 151 dos 256 possíveis *opcodes*, organizados em 56 instruções com diferentes modos de endereçamento.

O *6502* tinha uma linguagem de montagem composta por mnemônicos de instruções de três caracteres, seguidos por operandos quando necessário.

Tabela 1 – Especificações Técnicas do *MOS 6502*

Parâmetro	**Valor**
Arquitetura	*CISC 8-bit*
Frequência de *Clock*	1.023 MHz (*MOS 6502*) / 2 MHz (*MOS 6502A*)
Registradores	A (Acumulador), X, Y, P (*Status*), S (*Stack Pointer*), PC (*Program Counter*)
Tamanho de Palavra	8 *bits*
Endereçamento	16 *bits* (*64KB* de espaço de endereço)
Modos de Endereçamento	Absoluto, *Zero Page*, *Implied*, Indireto, Relativo, entre outros
Instruções	Mais de 50 instruções diferentes
Barramentos	Barramento de Dados de 8 *bits*, Barramento de Endereço de 16 *bits*

3.2 Visão externa

Abaixo, uma imagem da visão externa do processador fabricado pela própria MOS Technology no ano de 1975 (TECHNOLOGY, 1975), no modelo de quarenta pinos.

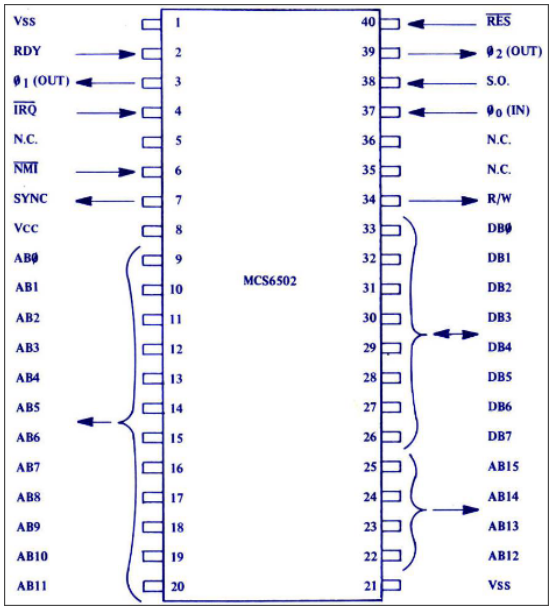

Figura 1 – Esquema visual do microprocessador

Como é possível perceber pela imagem, existem diversas portas de entrada e saída no dispositivo. Analisando essas portas, pode-se obter algumas informações interessantes sobre o componente, como: o número e espaço de endereçamento, sinais de *clock*, tipos de E/S e até os barramentos de dados.

Entre os números de 1 até 8, temos: o VSS (*Voltage Supply Ground*, que refere-se ao campo linha de terra de energia), o RDY (*Ready*, sinal que indica que o processador está pronto para aceitar operações), o IRQ (*Interrupt Request*, sinal usado para solicitar interrupções ao dispositivo), o NMI (*Non-Maskable Interrupt*, sinal de interrupção com prioridade), o SYNC (*Synchronization*, sinal que pode ser utilizado para sincronizar outros aparelhos com o microprocessador) e o VCC (*Voltage Common Collector*, que refere-se ao campo de alimentação positiva do componente).

Entre os números de 9 até 25, temos as áreas de endereçamento de memória, os símbolos AB0-AB15 (*Address Bus*, indexados de 0 a 15). Essas linhas podem ser utilizadas para endereçar componentes externos ao microprocessador, permitindo operações entre dispositivos de entrada e saída, bem como a escrita e leitura de dados. É importante notar que existem 16 linhas de endereçamento, motivo pelo qual o aparelho suporta até 64 KB de memória endereçável.

Entre os números de 26 até 33, temos as áreas de linhas de dados do microprocessador, os símbolos DB0-DB7 (*Data Bus*, indexados de 0 a 7). Essas linhas são utilizadas para transmitir dados entre componentes externos, facilitando operações de escrita, leitura, E/S e comunicação entre periféricos. Novamente, nota-se que existem 8 linhas de dados, indicando que o microprocessador é capaz de transmitir, se utilizadas simultaneamente, 8 *bits* nos barramentos.

Por fim, entre os números 34 até 40, temos algumas entradas auxiliares importantes, como: o R/W (*Read/Write*, sinal que indica se a operação é de escrita ou leitura), o S.O (*Serial Output*, que permite a saída de dados para comunicação serial) e o RES (*Reset*, sinal utilizado para reiniciar o dispositivo).

Existe também, na imagem, entradas chamadas de N.C., que significam *Not Connected*. No contexto de eletrônica, essa sigla indica que o pino ou linha não possui conexão a outro componente do circuito, ou seja, não possui uma função específica atribuída.

4 Exemplo de algoritmo

4.1 Busca binária

```
; Constantes
VALUE_TO_FIND = $FF         ; O valor que você quer encontrar
ARRAY_START = $2000         ; Endereço do começo do array ordenado
ARRAY_END = $20FF           ; Endereço do fim do array ordenado
ARRAY_LENGTH = $0100        ; Tamanho do vetor
; Variaveis
low = $00                   ; Índice da esquerda do intervalo de pesquisa
high = $00                  ; Índice da direita do intervalo de pesquisa
mid = $00                   ; Índice do meio do intervalo de pesquisa
found = $00                 ; Variável para indicar se o valor foi achado ou não
    LDA #VALUE_TO_FIND      ; Carrega o valor para encontrar
    STA found               ; Inicializa a variável ‘found‘
    LDA #ARRAY_START        ; Carrega o endereço do começo do vetor
    STA low
    LDA #ARRAY_END          ; Carrega o endereço do fim do vetor
    STA high

searchLoop: ; Loop da busca binaria
    CMP low, high           ; Compara a parte da direita com a da esquerda
    BCC notFound            ; se ‘low > high‘, o valor não está no vetor
    ; Calcula o indice do meio
    SEC
    SBC low, high
    LSR
    CLC
    ADC low, high
    LSR
    STA mid
    ; carrega o valor do indice do meio, da memoria
    LDA (mid)
    CMP #VALUE_TO_FIND      ; Compara com o valor para encontrar
    BEQ foundValue          ; Se eles são iguais, encontramos o valor
    ; Update search range
    BCC updateLow           ; se o valor do meio é menor que o valor pra pesquisar,
    updateHigh:             ; atualiza o indice da esquerda
        LDA mid
        INX                 ; Incrementa o indice do meio
        STA low             ; Atualiza o indice da esquerda para o meio +1
        BCC searchLoop
    updateLow:
        LDA mid
        DEX                 ; Decrementa o incide do meio
        STA high            ; Atualiza o indice da direita para o meio -1
        BCC searchLoop

foundValue: ; Valor encontrado
    LDA #1                  ; altera o ‘found‘ para indicar que o valor foi encontrado
    STA found
notFound: ; O programa vai pular aqui se não encontrar o valor procurado
    BRK ; Fim do programa
.org $2000 ; Dados do array
```

Abaixo, temos a explicação completa do código:

1. Constantes e variáveis
   - VALUE_TO_FIND: O valor que está sendo procurado no array. Neste caso, é definido como FF.
   - ARRAY_START: O endereço de memória onde começa o array ordenado.
   - ARRAY_END: O endereço de memória onde termina o array ordenado.
   - ARRAY_LENGTH: O tamanho do vetor (256 elementos, pois 0100 em hexadecimal é 256).
   - *low, high, mid*: Índices usados para definir o intervalo de busca no array.
2. Inicialização
   - LDA #VALUE_TO_FIND: Carrega o valor que está sendo procurado no registrador acumulador.
   - STA found: Inicializa a variável 'found' com o valor carregado.
   - LDA #ARRAY_START, STA low: Inicializa o índice da esquerda (*low*) com o endereço inicial do *array*.
   - LDA #ARRAY_END, STA high: Inicializa o índice da direita (*high*) com o endereço final do *array*.
3. Loop de busca binária (*searchLoop*)
   - CMP low, high: Compara os índices da esquerda e da direita.
   - BCC notFound: Se *low* ¿ *high*, o valor não está no vetor e o programa vai para a etiqueta *notFound*.
4. Cálculo do índice do meio
   - SEC, SBC low, high: Subtrai *low* de *high* com *carry*.
   - LSR, CLC, ADC low, high, LSR: Divide o resultado por 2, obtendo o índice do meio do intervalo de busca.
   - STA mid: Armazena o índice do meio na variável 'mid'.
   - LDA mid: Carrega o valor do *array* no índice do meio.
5. Comparação e atualização dos limites de busca
   - CMP #VALUE_TO_FIND: Compara o valor do *array* no índice do meio com o valor que está sendo procurado.
   - BEQ foundValue: Se são iguais, o valor foi encontrado e o programa vai para a etiqueta foundValue.
   - BCC updateLow: Se o valor no meio é menor que o valor procurado, atualiza o índice da esquerda.
   - updateHigh: Se o valor no meio é maior que o valor procurado, atualiza o índice da direita.
6. *Labels foundValue* e *notFound*
   - foundValue: Se o valor é encontrado, a variável 'found' é setada para 1.
   - notFound: Se o valor não é encontrado, o programa vai para esta *label* e então termina (BRK).

5 Peculiaridades e Falhas

A arquitetura do *6502* possuía alguns problemas e peculiaridades que precisavam ser levadas em consideração ao programá-lo.

A maioria dessas questões foram corrigidas nas versões *CMOS* do *6502*, proporcionando maior estabilidade e previsibilidade no processamento de instruções e interações com o *hardware*.

1. Falta de instrução *ROR*

   - As primeiras versões do *6502* não possuíam a instrução *ROR* (rotação à direita na memória ou acumulador). Esses chips executavam a operação de forma semelhante a *ASL* (deslocamento aritmético à esquerda) sem afetar o *bit* de transporte no registro de status.

2. Instruções não documentadas

   - A família *NMOS 6502* tinha várias instruções não documentadas que variavam entre os fabricantes de chips, resultando em comportamentos estranhos e imprevisíveis. Alguns programadores criaram dispositivos para lidar com essas instruções, enquanto outros as usaram para estender o conjunto de instruções.

3. Falha na instrução *JMP* indireta

   - A instrução de salto indireto na memória *JMP* (endereço) estava parcialmente com defeito, resultando em saltos incorretos quando o endereço terminava em FF.

4. Problemas com endereçamento cruzando limites de página

   - O *6502 NMOS* fazia uma leitura adicional em um endereço inválido ao cruzar os limites de página, o que poderia causar problemas inesperados.

5. Flags de status em modo *BCD (Binary Coded Decimal)*

   - Em modo *BCD*, as bandeiras de status N (negativo), V (*overflow* de *bit* de sinal) e Z (zero) eram geralmente irrelevantes, pois refletiam o resultado binário, não *BCD*.

6. Instrução *BRK*

   - A instrução *BRK*, que era uma interrupção de *software*, poderia ser interrompida por uma interrupção de *hardware* no *6502 NMOS*.

7. Endereço de retorno *JSR/RTS*

   - Os endereços de retorno empilhados pelo *JSR* eram do último *byte* do operando *JSR*, não do próximo após a instrução.

8. Atraso na leitura de *CPU*

   - A *CPU* podia ter seu acesso à memória atrasado temporariamente definindo o pino *RDY* como baixo, exceto durante o acesso de gravação, que só era atrasado na próxima leitura.

6 Cenário atual

Em vista dos fatos comentados, fica claro que o processador *MOS 6502* desempenhou um papel crucial na história da computação e nos primórdios da indústria de jogos, com relevância até os dias de hoje. Através de um design simples e eficiente, ele tornou a computação acessível para um público mais amplo e abriu portas para o desenvolvimento de software e jogos que transformaram toda a indústria. Ao longo desse trabalho, foram exploradas a história, arquitetura e características técnicas do dispositivo, destacando sua estrutura de registradores, uso de certas instruções e endereçamento.

Até os dias atuais, o *6502* continua uma peça icônica, sendo utilizado por muitos entusiastas de tecnologia, especialmente por aqueles que apreciam programação em baixo nível ou estão dispostos a aprender como a arquitetura do modelo funcionava nas máquinas antigas.

É possível encontrar frequentemente usuários que criam emuladores de sistemas que usavam o *6502*, como o *Commodore 64* e o *Apple II*. Dentre alguns simuladores, é possível encontrar vários na *web*, como o *Visual 6502* (GRIFFINI, 2012) feito em *JavaScript*, que permite acompanhar visualmente a simulação do funcionamento do dispositivo.

Na área de sistemas embarcados, também é muito utilizada a versão *CMOS* do *6502*, o *WDC 65C02*. O *WDC 65C02* é uma versão aprimorada do seu antecessor, consumindo menos energia, com menos falhas e com novas instruções. A melhoria no gasto de energia é tanta que o *WDC 65C02* chega a consumir até vinte vezes menos que o antigo *6502*, tornando-o um ótimo dispositivo para embarcados, onde a eficiência energética é crítica.

Referências

BRIGHT, B. *Assembly Language for the 6502: Pocket Guide*. [S.l.]: Pitman, 1983. (Programming pocket guides). ISBN 9780273019909.

CARR, J. J. *6502 User's Manual*. Reston, Virginia: Reston Pub. Co., 1984.

FOSTER, C. C. *Programming a Microcomputer: 6502*. Reading, Massachusetts: Addison-Wesley Pub. Co., 1978.

GRIFFINI, A. *Javascript 6502 emulator*. 2012. Disponível em: ⟨http://www.visual6502.org/JSSim/⟩.

TECHNOLOGY, M. *MCS6501 - MCS6505 Microprocessors Datasheet*. 1975. Disponível em: ⟨http://archive.6502.org/datasheets/mos_6501-6505_mpu_preliminary_aug_1975.pdf⟩.