MICROCONTROLADORES E MICROPROCESSADORES: UM ENFOQUE PRÁTICO

Mauro Conti Pereira – maurocp@gmail.com
Edson Antonio Batista – edson.ucdb@gmail.com
Wanderlei Mendes Ferreira – mendes.ucdb@gmail.com
Hemerson Pistori – pistori@ucdb.br
Universidade Católica Dom Bosco, Cursos de Eng.Mecatrônica, Mecânica e de Computação
Grupo de Pesquisa em Engenharia e Computação
Av. Tamandaré 6000, Jardim Seminário
79117-900 – Campo Grande, MS.
Alexandre César Rodrigues da Silva – acrsilva@dee.feis.unesp.br
Depto de Engenharia Elétrica, Faculdade de Engenharia de Ilha Solteira, UNESP
Ilha Solteira, SP.

Resumo: *Este artigo descreve um enfoque prático para o ensino de microcontroladores em nossos cursos de engenharia de computação e engenharia mecatrônica. Os estudantes chegam com conhecimento de eletrônica digital e analógica de disciplinas anteriores, e durante o semestre têm aulas teóricas e práticas, que cobrem noções de arquitetura de computadores para entender o funcionamento interno dos processadores envolvidos, circuitos de interface para funcionarem e interagirem com o mundo real, e programação em linguagem montadora para realizarem tarefas específicas. Os estudantes são divididos em grupos, cada um escolhendo um projeto final a ser realizado, envolvendo implementações de hardware e software. No correr do semestre, pequenos projetos de hardware são submetidos, que ajudarão a compor o projeto final. Isto tem motivado os alunos a aprenderem e testarem suas próprias idéias, melhorando sua iniciativa e confiança.*

Palavras-chave: *Microcontroladores, Arquitetura de computadores, Sistemas Microprocessados, Aulas Práticas.*

1 INTRODUÇÃO

Na maioria das disciplinas do curso de Engenharia de Computação da UCDB, e posteriormente no de Mecatrônica, sempre há a preocupação de equilíbrio entre os aspectos de hardware e software, e teoria e prática na formação de nossos egressos. Sempre existe uma necessidade de exigir implementações práticas dos conceitos teóricos vistos, seja em disciplinas de hardware ou de software. Assim, diversas disciplinas foram projetadas para

terem aulas teóricas e de laboratório, estas últimas servindo de motivação aos alunos e reforço para fixação dos conceitos vistos concomitantemente nas aulas teóricas. Entre outras disciplinas que são trabalhadas desta maneira podem-se citar algumas que envolvem conceitos de eletricidade, eletrônica básica, eletrônica digital, arquitetura de computadores, redes de computadores e noções de telecomunicações.

Estas disciplinas usam também avaliações práticas e projetos integradores. Continuando com a mesma filosofia, também na disciplina de Microcontroladores isto foi buscado, pois isto tem demonstrado que fomenta autonomia, iniciativa e confiança em nossos egressos. Procuramos assim passar os conceitos básicos de arquitetura de computadores, circuitos de interfaceamento e programação básica, buscando que os alunos implementem alguma solução, desde os circuitos de acionamento (*reset, clock,* etc...), placas de circuito impresso, até o software final que acione o sistema completo. Isto serve para que o aluno possa simular um ambiente de trabalho, buscando soluções e desenvolvendo um sistema que resolva um problema levantado por ele. Os diversos aspectos descritos em aulas teóricas e práticas são comentados a seguir.

A disciplina é dada em forma de 6 aulas semanais de 50 minutos, em três dias, sendo dois dias de 100 minutos de teoria, e mais um de 100 minutos para a prática, que inclui desde montagem de circuitos até programação e depuração de programas, mas com a maior parte das implementações sendo feitas fora da aula. Se a turma total for grande, dividem-se as turmas de laboratório em grupos de no máximo 20 alunos cada. São 18 semanas por semestre, sendo usado um semestre apenas para a disciplina completa.

Para ganhar tempo procura-se usar aulas prontas e ferramentas como *datashow* e transparências para apresentar a teoria, permitindo que sobre mais tempo para resolução de exemplos e exercícios dirigidos, ou para a implementação de circuitos nas aulas práticas.

2 AULAS TEÓRICAS

Inicia-se com o estudo de noções de arquitetura de computadores, usando exemplos de programa em linguagem de máquina e *assembly*, incrementalmente passando os conceitos de arquitetura e programação desta arquitetura. A cada incremento novas idéias são adicionadas, novos registradores, novos conceitos e novas instruções. Chega-se até a mencionar, sem uso de maiores formalismos, mas com uso de analogias a conhecimentos do dia-a-dia, conceitos mais avançados como memória *cache*, *bit-slice,* paralelismo, arquitetura Von Neuman x Harvard, e arquitetura RISC x CISC.

Numa etapa seguinte são passados conceitos de processadores comerciais, algumas famílias mais conhecidas são citadas, como Intel, Motorola, DSPs da Texas Instruments, inclusive com estudo mais detalhado de alguns *chips* de controle de periféricos, como portas paralelas, seriais, e controlador de vídeo, DMA e interrupções tais como 8255, 8251, 6845, 8257, 8259, e até conversores AD/DA como TMS320C40, entre outros.

Escolhe-se então uma família especifica de microcontroladores, tais como os da Microchip e Atmel, para o desenvolvimento do projeto final e desenvolvimento das aulas de programação.

Os alunos chegam a esta disciplina com conhecimento de circuitos digitais, tanto combinatórios como seqüenciais, envolvendo inclusive máquinas de estados finitos e todos os conceitos práticos, como famílias, níveis de tensão, *fan-in/fan-out, latches, buffers,* e assim por diante. Isto facilita a compreensão com visão *botton-up* de arquitetura de computadores, que é introduzida a partir de versões simplificadas de arquiteturas, com poucos registradores e modos de endereçamento, além de barramentos estreitos, inclusive um de endereços com apenas 8 bits. Baseia-se numa arquitetura simplificada de um microcontrolador popular, o 8051. Entre os registradores básicos destacam-se inicialmente apenas alguns registradores essenciais, PC *(program counter),* AR *(address register),* IR *(instruction register),* A

(acumulador), B (registrador auxiliar). Inicialmente se faz uso de algumas poucas instruções como carregamento imediato e soma, depois são acrescentados outros a cada novo exemplo com instruções mais complexas e com mais modos de endereçamento (indireto, indexado, etc...). Posteriormente são passados exemplos envolvendo a chamada de sub-rotinas com uso do SP (*stack pointer*).

Para simular o funcionamento desta arquitetura, faz-se a execução passo a passo de cada instrução, destacando-se os ciclos de busca e execução, mostrando num quadro a evolução do conteúdo dos registradores. Assim, para executar o programa, o aluno preenche o conteúdo de cada registrador após cada etapa de micro-operação da instrução. É fornecido o conteúdo de uma área da memória com o programa, em hexadecimal (Tabela 1), um quadro com descrição das instruções e seus hexadecimais e mnemônicos (Tabela 2), e o quadro de execução dos conteúdos dos registradores (Tabela 3).

Alguns livros de Arquitetura de Computadores utilizam o mesmo expediente de detalhar arquiteturas simplificadas e ir progredindo aos poucos, crescendo a complexidade pouco a pouco, mas deixando para que o aluno implemente a arquitetura (GAJSKI, 1997), (BLAAW,1997), (FREGNI e SARAIVA, 1994), ou use algum simulador (WEBER, 2001). Isto dificulta a compreensão do aluno, pois o nível de abstração fica mais elevado. Com o uso do quadro de execução da Tabela 3, pode-se dispensar o uso de simulador, com o aluno tomando para si a tarefa de mover os conteúdos dos registradores para buscar a instrução, decodificá-la e executá-la, conforme exemplo a seguir. Por simplificação, os endereços são de apenas 8 bits, i.e., 256 posições endereçáveis, e são mostrados apenas quando se mudam os valores dos registradores.

Tabela 1. Conteúdo de memória com códigos hexadecimais a serem decodificados e executados manualmente.

Posição	Conteúdo inicial	Conteúdo final
51	23	
50	22	
49	21	
48	00	
…	…	
32	76	
31	75	
30	FE	
2F	1 A	
2E	2D	
2D	2B	
2C	19	
2B	76	
2A	1B	
29	2C	

Posição	Conteúdo inicial	Conteúdo final
28	74	
27	FE	
26	1 A	
25	2D	
24	2B	
23	19	
22	2D	
36	76	
...	...	
1B	00	1E
1A	06	
19	05	
18	00	
17	00	

Este método tem a vantagem de permitir a visualização da evolução dos conteúdos dos registradores de uma só vez, para reavaliar a execução do programa, algo que simuladores não permitem. Ele permite ainda que se fixe bem cada ciclo das instruções, perfeitamente a busca, decodificação (feita pelo próprio aluno) e execução da cada uma de suas micro-operações que compõem a instrução. Usado no ensino de microcontroladores e microprocessadores, ele

facilita a compreensão das operações internas dos registradores. Se ele for usado no ensino de Arquitetura de Computadores permite que se encarregue o aluno de fazer pequenos projetos incrementais, permitindo descobrir necessidades de conexões ou mesmo de novos registradores auxiliares, facilitando o entendimento e projeto dos barramentos internos da arquitetura em questão.

Tabela 2. Quadro para decodificar o código fonte (*aa* indica endereço e *dd* indica dados).

Operação	operando	código	DESCRIÇÃO
CALL	Label	3E aa	INC SP, PC=>[SP], aa=>PC
MOV	A,[aa]	2D aa	Copia conteúdo do endereço aa => Acumulador (endereçamento indireto)
MOV	B,[aa]	2E aa	Copia conteúdo do endereço. aa => B
MOV	[aa],A	2C aa	Copia conteúdo do acumulador p/o endereço aa
MOV	B,A	2B	Copia conteúdo do acumulador A p/o registrador B
MOV	A,#dd	4A dd	Copia conteúdo seguinte indicado pelo PC para o registrador A (end.imediato)
HLT		76	Pára o processamento
NOP		74	NADA
INC	A	1A	A + 1 => A
SHR	A	DD	Shift right: 0=>b7, b7=>b6,... b2=>b1, descarta b0
MUL	A,B	FE	A * B => A
RET		75	[SP]=>PC, DEC SP
DEC	A	00	A - 1 => A

Observe que a primeira coluna do quadro de execução da Tabela 3 e 4 mostra os valores iniciais dos registradores antes do inicio da execução, e a última coluna mostra os valores finais dos registradores após a execução do código.

Tabela 3: Quadro de execução das instruções, passo a passo. (primeira parte)

		MOV A,[19]					MOV B,A			MOV A,[1B]				
		busca		Execução			busca		ex	Busca		Execução		
PC	22	23		24			25			26		27		
AR	XX	22		23	19		24			25		26	1A	
IR	XX		2D					2B			2D			
SP	48													
A	00					05								06
B	00								05					
C	00													
Endereço														
Conteúdo do end.														

Tabela 4: Quadro de execução das instruções, passo a passo. (segunda parte)

	MUL A,B		NOP	MOV [1B],A		HLT
	Busca	ex	busca	Busca	Execução	busca

PC	28			29		2A		2B			2C	**2C**
AR	27			28		29		2A	1B		2B	**76**
IR		FE			74		2C					**2C**
SP												**XX**
A			1E									**1E**
B												**05**
C												**00**
Endereço										1B		
Conteúdo do end.										1E		

Também são revistos e ampliados conceitos de eletrônica digital, especificamente a parte de decodificação de endereços, que se iniciou com a associação de memórias. É visto como se decodificam os endereços para interligar microprocessadores a memórias, controladores de periféricos, displays LCD e dispositivos semelhantes. Utiliza-se o mapa de endereços e geração dos sinais de *chip select* dos periféricos conforme mostra o quadro da Tabela 5. O exemplo refere-se a um microprocessador de 8 bits e 16 linhas de endereçamento com os seguintes endereços reservados:

- 0 a 8K-1: memória ROM
- 8K a 16K -1: memória RAM estática
- 16K a 24K-1: memória RAM estática
- 32K em diante: 4 posições para mostrador LCD

Tabela 5. Mapa de decodificação de endereços

End. decimal	Hexa-Decimal	Linhas de endereço		Linhas de dados	*Chip selects* (ativos em 0)			
		$A_{15}A_{14}A_{13}$	$A_{12}.....A_0$	D7....D0	ROM	RAM0	RAM1	LCD
0 ... 8K-1	0000 ... 1FFF	000	0 0000 0000 0000 ... 1 1111 1111 1111	ROM	0	1	1	1
8K ... 16K-1	2000 ... 3FFF	001	0 0000 0000 0000 ... 1 1111 1111 1111	RAM0	1	1	1	1
16K ... 24K-1	4000 ... 5FFF	010	0 0000 0000 0000 ... 1 1111 1111 1111	RAM1	1	1	1	1
24K ... 32K-1	6000 ... 7FFF	011	0 0000 0000 0000 ... 1 1111 1111 1111	livre	1	1	1	1
32K ... 32K+3 ... 40K-1	8000 ... 8003 ... 9FFF	100	0 0000 0000 0000 ... 0 0000 0000 0100 ... 1 1111 1111 1111	LCD	1	1	1	1
40K ... 64K-1	A000 ... FFFF	101 ... 111	0 0000 0000 0101 ... 1 1111 1111 1111	Livre	1	1	1	1

A Tabela 5 permite ao aluno visualizar melhor o porquê dos circuitos de decodificação, e a partir dela é possível então escrever as equações lógicas dos pinos de ativação dos periféricos a partir das linhas de endereço não usadas internamente nas memórias, escolhendo

então o uso de decodificadores prontos como 74138 ou 74139, ou permite desenvolver circuitos com lógica reprogramável mais simples com famílias *PAL* e *GAL*, conforme é bem descrito em (CARTER, 1997).

Adicionalmente, para projetar os circuitos de hardware, especificamente os circuitos de *clock, reset,* acionamento e interface de motores, reles, teclados, alto-falantes e similares, são necessários conhecimentos de eletrônica analógica, como carregamento de capacitores, divisores de tensão, osciladores, amplificadores operacionais e assim por diante, tudo visto em disciplinas anteriores, mas revisado rapidamente em cada tópico abordado. Isto não precisa ser revisto aqui por ser facilmente encontrado e tradicionalmente coberto em literaturas especificas como (ARTWICK, 1980), [CADY, 1997], (CIARCIA,1984) ou (MIMS, 1992). Contudo, é conveniente simular com *PSPICE* ou *Electronics Workbench* (atual *Multisim*), como visto em (MELO,1998), (ADAMS,2000) e (ALBUQUERQUE,1998). Isto visa estimulá-lo a tentar entender e projetar o circuito antes de testá-lo realmente, agilizando a depuração e o aprendizado.

3 AULAS PRÁTICAS

Nas aulas práticas cobra-se que os alunos calculem, projetem e implementem os circuitos vistos em teoria, dando ênfase que a maior parte do trabalho seja feito fora do horário de aula.

Como mencionado, na teoria são revistos alguns conceitos básicos de disciplinas anteriores, para que se possam gerar circuitos como os de interface, de *clock e reset,* entre outros.

Apresentamos também, em uma aula prática, os conceitos de como implementar os circuitos em placas de circuito impresso (PCI) em vez de apenas no *protoboard*, para que se possam montar circuitos mais elaborados e próximos de algo que se faria na vida real. Por simplificação, utilizam-se circuitos em encapsulamento DIP (*dual in-line package)*, por sua maior facilidade de obtenção e de uso. Isto é algo que normalmente não se ensina em escolas de engenharia, mas que avaliamos ser útil para melhorar o desenvolvimento e motivar os alunos, já que é gasto apenas um dia (aula dupla com 100 minutos) para passar os conceitos básicos de como se fazer uma PCI razoável, e de implementá-la. Um bom livro que cobre o assunto, bem como a parte de confecção de caixas e documentação de projeto é (REIS, 2004). Considerações de construção de projetos incluem invólucro (caixas), fonte de alimentação, construção de PCI, compra de componentes, tipos de soquetes de CPU: DIP, *grid array*, PLCC, QFP, TQFP, BGA, *Slot* e montagem e teste.

Procura-se também comparar pequenas soluções microprocessadas com sistemas seqüenciais síncronos resolvidos de maneira clássica usando portas e *flip-flops* quaisquer, depois usando EPROM e flip-flop tipo D, depois com PAL/GAL e finalmente com FPGAs, usando-se de ferramentas e diagramas de maior facilidade na transição entre o modelo e a implementação.

Estes pequenos detalhes têm demonstrado uma maior eficiência no aprendizado do que a tradicional maneira de se introduzir diversos conceitos teóricos de difícil apreensão por parte dos alunos, e somente depois fazer exemplos que esclareçam tais exemplos.

Para o ensino da linguagem *assembly* são utilizados os métodos e ferramentas tradicionais encontrados em livros como [MACKENZIE, 1998] e (SILVA,1990), incluindo kits didáticos de microprocessadores de empresas nacionais como Hidroeletro, CNZ, Datapool, Mosaico, ou norte-americana como a AES (Advanced Educational Systems).

Os programas também são incrementados pouco a pouco incluindo paulatinamente conceitos de novos modos de endereçamento, desvios, sub-rotinas, macros e comparações, através do desenvolvimento de rotinas para implementação de uma pequena calculadora de quatro operações no kit utilizado, que vem com LCD (*liquid cristal display)*, teclado e interface.

Numa segunda etapa, desenvolvem-se os programas para o projeto especifico que cada grupo escolheu.

São cobertos tópicos tais como:

- Linguagem de máquina x de montagem
- Linha padrão*: label*, *opcode*, *operand*, *comment*
- Montadores 1-*pass* x 2-*pass*
- Código fonte x objeto x executável
- Listagens: *cross reference*, mapas de memória
- *Symbol tables*: endereço simbólico, caracteres (símbolos reservados, *user-defined*, *asm generated*, *global*, *absoluto*, *relocatable*)
- Formato de *Source line*
- Conjunto de caracteres delimitadores, *label*, *opcode*, *operand*, *comment*
- Avaliação de expressões em tempo de montagem: operadores, valores permitidos, precedência, expressões relocáveis
- Instruções e diretivas
    - *Set* de instruções
    - Definições de símbolos e de dados (QUE, SET, DB, DW)
    - Origem e reserva de memória e terminação ((ORG, DS, END)
    - Montagem condicional (IF, ELSE, ENDIF...)
- Macros
    - Macros x sub-rotinas
    - Diretivas de macros (MACRO, ENDM, LOCAL, REPT, IRP, IRPC, EXITM)
    - *Nesting* (aninhamento)
- Estruturas de dados em *assembly*
    - *Branch tables*
    - Transferência de dados para sub-rotinas
    - FIFO, LIFO, *lookup tables*, *stacks*
- DEPURAÇÃO (*DEBUGGING*)
    - Conceitos de depuração de programas *assembly*
    - *Breakpoint*
    - *Tracing*
    - *Step-by-step*
- Ferramentas de desenvolvimento
    - Ferramentas de software: *Assemblers* e *linkers*, Simuladores
    - Ferramentas de hardware: Emulador ICE (In-C*ircuit Emulation*), ISP (*In-System Programming),* JTAG
- Métodos de I/O
    - Endereçamento de I/O: dedicado x *memory mapped*
    - *Polling* x interrupções
    - Comunicação de dados:
        - paralela,
        - serial: *baud rate*, síncrona x assíncrona, protocolos por sw ou por hw.
        - DMA

No início do semestre letivo, os alunos são divididos em grupos e escolhem um circuito microprocessado a ser realizado totalmente, por exemplo, um mini-orgão ou um brinquedo do tipo *Genius*, em que se devem repetir teclas acionadas aleatoriamente pelo processador, emitindo diferentes tons para cada uma das quatro teclas.

A cada semana, etapas são realizadas, de hardware e de software, com *deadlines* intermediários a serem cumpridos, que contam para a nota final. Se a equipe não terminar o projeto, fica "de exame", mas com o exame da disciplina sendo terminar o projeto.

4 CONSIDERAÇÕES FINAIS

A partir da implementação das técnicas aqui descritas, foi obtido sucesso aumentando cada vez mais o interesse e o aproveitamento dos alunos. Revisar a teoria antes de pedir o projeto de cada pequena parte do hardware se provou eficiente. Também destacamos que o uso de ferramentas para apresentar a teoria disponibilizou mais tempo para exercícios, contribuindo para melhor fixação dos conceitos.

Para o laboratório, com os alunos simulando previamente, possibilitou-se um melhor aproveitamento do tempo destinado às aulas práticas. O uso de simuladores e depuradores dos microcontroladores escolhidos agilizou a depuração e aprendizagem. Com o projeto final, aumentou-se a procura pelos professores para esclarecer dúvidas, aumentando a utilização do laboratório em seus horários antes livres. A participação em aula aumentou, o interesse também, e muitos alunos passaram a desfrutar mais a parte prática, aumentando sua iniciativa e autoconfiança.

5 REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, R.O. **Análise e Simulação de Circuitos no Computador**: EWB5, Érica, São Paulo, 1998.

ADAMS, J. ***Mastering Electronics Workbench***. New York: McGrawHill, 2000.

BLAAW, G.A. BROOKS Jr., F.P. ***Computer Architecture: Concepts and evolution***. Reading,MA: Addison Wesley, 1997.

CARTER, J. ***Digital Designing with Programmable Logic Devices***. Englewood Cliffs, NJ: Prentice Hall, 1997.

FREGNI, E.; SARAIVA, A.M. **Engenharia do projeto lógico digital**, São Paulo: Edgard Blucher, 1994.

MELO, J. **SPICE: Simulando projetos eletrônicos no computador**, São Paulo: Érica, 1998.

SILVA, V.P. **Aplicações Práticas do Microcontrolador 8051**. São Paulo: Érica, 1990.*

GIMENES, S.P. **Microcontroladores 8051**. São Paulo, Prentice Hall Brasil, 2002.

NICOLOSI, D.E.C. **Laboratório de microcontroladores família 8051: treino de instruções, hardware e software**. São Paulo: Érica, 2004

ADVANCED EDUCATIONAL SYSTEMS. **Manuais do kit didático AES-51**. Santa Ana, CA: Advanced Educational Systems, 1996.

ARTWICK, Bruce A. ***Microcomputer Interfacing***. Englewood Cliffs, NJ: Prentice Hall. 1980.

CADY, F.M. ***Microcontrollers and Microcomputers: Principles of software and Hardware Engineering.*** Oxford: Oxford University Press, 1997.

CIARCIA, S. ***Build your own Z80 computer***. New York: McGrawHill, 1984.

GAJSKI, D.D. ***Principles of Digital Design***. Englewood Cliffs, NJ::Prentice Hall, 1997.

MACKENZIE, I.S. **The 8051 Microcontroller.** Upper Saddle River, NJ: Prentice Hall, 1999.
OSBORNE, A.

MIMS, F. ***Engineers Notebook***. Solana Beach, CA: Hightext Publications, 1992.

WEBER, R.F. **Fundamentos de arquitetura de computadores** Porto Alegre: EdSagra Luzzato, 2ed. 2001.

MICROCONTROLLERS AND MICROPROCESSORS: A PRACTICAL APPROACH

Abstract: *This paper describes a practical approach to teaching microcontrollers in both computer and mechatronics engineering program at UCDB, in Brazil. The students come with knowledge of digital and analogic electronics from previous classes, and during the semester they have theoretical and practical lessons that cover notions of computer architecture to understand the inner workings of the processors involved, interface circuits to make it work and interact with the real world, and assembly language programming to perform dedicated tasks. The students are divided into teams, which choose a final project to be done, involving hardware and software implementations. During the semester partial hardware projects and programming tasks are submitted that will be part of this final project. This has motivated the students towards learning and testing their own ideas, improving their initiative and confidence.*

Key-words: *Microcontrollers, Computer Architecture, Microprocessor Systems, Practical Classes*