NeanderWin -Um Simulador Didático para uma Arquitetura do Tipo Acumulador

3 authors, including:

José Antonio Dos Santos Borges
Federal University of Rio de Janeiro
37 PUBLICATIONS **55** CITATIONS
SEE PROFILE

Gabriel P. Silva
Federal University of Rio de Janeiro
33 PUBLICATIONS **24** CITATIONS
SEE PROFILE

NeanderWin - Um Simulador Didático para uma Arquitetura do Tipo Acumulador

José Antonio S. Borges
NCE/UFRJ e UNESA
antonio2@nce.ufrj.br

Gabriel P. Silva
DCC-IM/UFRJ
gabriel.silva@ufrj.br

Resumo

O uso de simuladores que permitam a compreensão do funcionamento do processador é fundamental para o ensino adequado de arquitetura de processadores. Este artigo apresenta um simulador de código livre para o Neander-X, que é uma extensão da conhecida arquitetura do Neander. O simulador aqui apresentado estende o conjunto de instruções inicialmente proposto e torna disponível um ambiente integrado de desenvolvimento, onde o aluno pode editar, compilar e executar código de programas escritos na linguagem de montagem do processador Neander-X.

1. Introdução

Um dos problemas encontrados no ensino de arquitetura de computadores é fazer com que os alunos compreendam corretamente o funcionamento de um processador, proporcionando também uma visão comparativa sobre algumas possibilidades arquiteturais. As fontes bibliográficas que são usadas no Brasil em particular [1], [2], [3] e [4] apresentam exemplos de arquiteturas relativamente complexas, talhadas para cursos dados no exterior sob condições de ensino ideais que incluem bons laboratórios de computação e monitoria. Essas estratégias de cursos nem sempre são de fácil aplicação, especialmente nas universidades brasileiras com menores recursos e com alunos de menor nível técnico.

Nas várias disciplinas que incluem aspectos de arquitetura de computadores, que ao longo dos anos temos aplicado em vários cursos e em vários níveis, muitas vezes nos deparamos com a necessidade do uso de modelos mais simples, não só quando se trata de disciplinas aplicadas nos primeiros períodos dos cursos de ciência da computação, mas também nos cursos de sistemas de informação, onde o uso de um modelo complexo pode significar grandes dificuldades na assimilação desses conceitos.

O ensino efetivo de arquiteturas de computadores praticamente obriga o professor ao uso de um simulador. Infelizmente, os simuladores atualmente disponíveis para ensino no Brasil (o que muitas vezes significa serem sistemas gratuitos), apresentam uma interface de usuário pouco elaborada e com poucos recursos operacionais, e mesmo considerando o uso de arquiteturas mais simples, é usual por parte dos alunos uma certa dificuldade de trabalhar com eles. Por exemplo, alguns simuladores exibem a necessidade da codificação do programa diretamente em linguagem de máquina; outros exigem a execução de seqüências de comandos para realizar as ações; outros ainda só suportam a execução em modo DOS.

Por outro lado, o ciclo de depuração de qualquer programa (em particular programas em linguagem de montagem ou linguagem de máquina) exige diversas modificações no código, com idas e vindas entre as etapas de codificação, compilação e execução. O resultado é que poucos alunos conseguem resolver as tarefas de uso do simulador com precisão, e existe sempre alto índice de cópia de soluções.

O sistema simulador NeanderWin, apresentado neste artigo, procura resolver esses problemas através um ambiente integrado de desenvolvimento, onde o aluno pode editar o código em linguagem de montagem, compilar e receber imediatamente mensagens relativas a erros de sintaxe, carregar na memória e simular a execução do programa, com visualização imediata e altamente interativa. O programa, por ser distribuído em código aberto, viabiliza a sua expansão (por outros professores ou por alunos em projeto), possibilitando a exploração de variantes da arquitetura ou adição de novas ferramentas de ensino ou projeto.

O NeanderWin se tornou uma ferramenta muito útil para o ensino de arquitetura de computadores tendo sido usada pelos autores em turmas com níveis acadêmicos diversificados sempre com grande aproveitamento dos alunos.

2. Trabalhos anteriores

Existem muitas ferramentas de simulação para processadores e microcontroladores comerciais. Os simuladores comerciais mais usados em projetos de equipamentos na atualidade (e que em princípio poderiam ser usados nos cursos básicos de arquitetura de computadores) são os de 8051, PIC e variantes de RISC. São sistemas completos, contendo muitas ferramentas integradas, mas quase todos, sendo destinados a uso profissional, têm licença de uso bastante cara e uso relativamente complexo, inviabilizando o uso amplo no ensino de graduação em muitos cursos no Brasil.

Na área didática também existem vários sistemas, como descritos em [5]. Os mais conhecidos sistemas gratuitos usados na pós-graduação no Brasil são os simuladores das arquiteturas do DLX [6] e MIPS64 [7], disponibilizados como material de apoio aos livros de Hennessy e Patterson, respectivamente na segunda e terceira edições. Esses sistemas são bastante completos como simuladores de arquitetura, mas não se mostram adequados para cursos introdutórios ou para cursos de curta duração, tanto pelas características da arquitetura quanto por detalhes operacionais do programa.

Um simulador interessante é o W-Neander, produzido como software companheiro do livro de Raul F. Weber, *Fundamentos de Arquitetura de Computadores* [8]. A arquitetura Neander é muito simples e pode ser explicada em uma ou duas aulas. Porém esse programa não tem montador integrado nem pode ser facilmente acoplado com outros módulos. Consideramos o conjunto de instruções do Neander muito limitado, sendo impossível criar problemas pouco mais do que triviais. Apesar destes problemas, foi usado com alguma reserva em um curso de segundo período, onde ficaram claras as dificuldades dos alunos em operá-lo.

Nossa idéia foi unir as estratégias de integração de ferramentas, encontradas nas ferramentas profissionais com uma arquitetura simples como a do Neander, que seria expandida para diminuir algumas limitações e ampliar o seu potencial para exercícios didáticos, consolidando a criação de um sistema que denominamos NEANDER-X

A definição desse sistema foi facilitada pela experiência anterior de um dos autores (Borges), no desenvolvimento de sistemas emuladores experimentais como o descrito em [9] e de um conhecido sistema CAD para ensino de Microeletrônica – TEDMOS, descrito com detalhe em [10]. Apesar de já obsoleto, esse sistema foi usado no passado para treinar centenas de alunos de engenharia eletrônica e pós-graduação no Brasil e exterior. O conhecimento de sua estrutura interna e de sua operação simples esclareceu muitos detalhes técnicos que foram usados na implementação do NEANDER-X.

3. Arquitetura do Neander-X

A máquina Neander como definida por Weber, é uma arquitetura rudimentar baseada em acumulador, de caráter didático, que pode ser completamente apresentada em uma ou duas aulas. Uma análise superficial do conjunto de instruções, entretanto, torna claro que muitas operações usuais (como chamada de rotinas, indexação, ponteiros, etc) são difíceis ou mesmo impossíveis de serem com ele implementadas. O mesmo livro apresenta outras arquiteturas mais sofisticadas, mas nenhuma realmente tão simples e com vantagens para o ensino em tempo curto.

Algumas características do processador original da máquina Neander incluem:

- Largura de dados e endereços de 8 bits;
- Dados representados em complemento a dois;
- 1 acumulador de 8 bits (AC);
- 1 apontador de instruções de 8 bits (PC);
- 1 registrador de código de condição com 2 bits: negativo (N) e zero (Z).

O Neander só possui um modo de endereçamento: o modo direto (absoluto), no qual a palavra que segue o código da instrução contém, nas instruções de manipulação de dados, o endereço de memória do operando. Nas instruções de desvio, esse endereço corresponde à posição de memória onde está a próxima instrução a ser executada.

A simplicidade da arquitetura permite seu ensino em tempo muito curto (duas aulas de 3 tempos) mas em pouco tempo as limitações aparecem e as necessidades de ensino deixam de ser alcançadas.

Para tornar possível atender aos requisitos mostrados no item 1 deste trabalho, mantendo a facilidade de ensino e sem fazer uso de outras arquiteturas, seria necessário estender o conjunto de instruções da máquina original para incluir alguns detalhes à arquitetura. A arquitetura estendida, denominada NEANDER-X, mostrada na Figura 1, incluiu, entre outros detalhes:

- A carga de dados imediatos no acumulador, simplificando operações de atribuição de dados;
- Um novo modo indireto de endereçamento, possibilitando exercitar as noções de indexação e ponteiros – que são básicas para entendimento de qualquer estrutura básica de programação;
- Operações de entrada e saída de dados para dois dispositivos mapeados em nosso simulador: um painel de chaves e um visor.

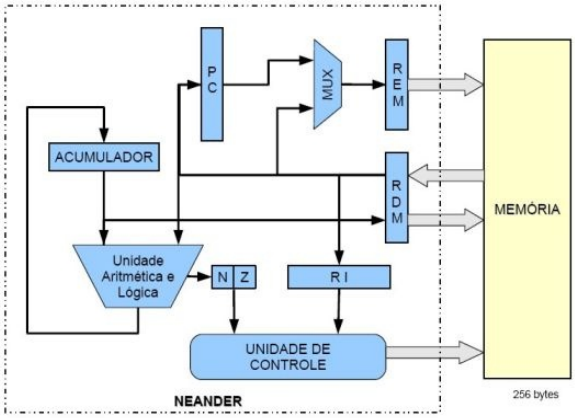

Figura 1 – Arquitetura do NEANDER-X

A Tabela 1 exibe o conjunto de instruções do NEANDER-X. As instruções podem ter um ou dois bytes (Figura 2). Nas instruções com apenas um byte, os 4 bits mais significativos contêm o código da operação. As instruções com dois bytes, são aquelas que fazem referência a um dado imediato ou operando na memória. Os 4 bits de mais baixa ordem do primeiro byte são reservados para futuras expansões.

Tabela 1 – Conj. de instruções do NEANDER-X

Cod.	Instrução	Descrição
0000	NOP	nenhuma operação
0001	STA ender	armazena acumulador (store)
0010	LDA ender	carrega acumulador (load)
0011	ADD ender	soma
0100	OR ender	operação lógica "ou"
0101	AND ender	operação lógica "e"
0110	NOT	operação lógica "negação"
0111	SUB ender	subtração
1000	JMP ender	desvio incondicional (jump)
1001	JN ender	desvio condicional (jump on negative)
1010	JZ ender	desvio condicional (jump on zero)
1011	JNZ ender	desvio condicional (jump on not zero)
1100	IN ender	operação de entrada no dispositivo "ender"
1101	OUT ender	operação de saída no dispositivo "ender"
1110	LDI imed	carrega o valor imediato "imed" no acumulador
1111	HLT	término da execução (halt)

Como mencionado anteriormente, existem três modos de endereçamento:

- Imediato – neste modo o segundo byte da instrução é o operando. A única instrução que usa este modo de endereçamento é a LDI.
- Direto – neste modo, segundo byte da instrução é o endereço de memória do operando.
- Indireto – neste modo, o segundo byte da instrução contém o endereço de memória onde está o endereço do operando (ou seja, o segundo byte da instrução é o endereço do ponteiro para o operando). Na linguagem de montagem, usou-se como convenção para indicar que um operando é indireto precedê-lo pela letra "@" (arrôba)

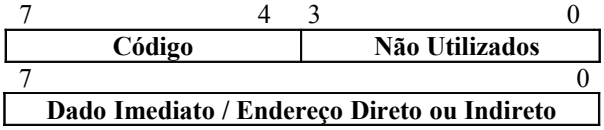

Figura 2 – Formato básico das instruções

4. A linguagem do montador NEANDER-X

Foi definida uma linguagem de montagem (assembly language) para este processador obedecendo a regras usualmente encontradas nos programas comerciais:

a) Formato geral das instruções

Uma linha pode conter alguns dos seguintes elementos: um rótulo, um operador ou uma pseudo-instrução, um operando opcional e comentários. São permitidas linhas vazias.

b) Comentários no programa

Os comentários são começados por ponto e vírgula, e podem também ocorrer no final das linhas de instruções.

c) Rótulos

Um rótulo é um nome dado à próxima posição de memória. O nome é seguido por dois pontos (a única exceção é a pseudo-instrução EQU).

d) Pseudo Instruções

ORG ender – ORG (origin) indica ao montador que a próxima instrução ou dado será colocado na posição **ender** de memória.

var EQU imed – EQU (equate) atribui um nome (rótulo) a um determinado valor. Entre muitos usos possíveis, esse comando pode ser usado para especificar variáveis que são posicionadas em um endereço específico de memória.

END ender – END indica que o programa fonte acabou. O operando **ender** é usado para pré-carregar o PC com um endereço inicial do programa.

DS imed – DS (define storage) reserva um número de palavras na memória definido pelo valor **imed**.

DB imed – DB (define bytes) carrega esta palavra com o valor dado pelo operando **imed**.

e) Representação de números

O número 48 teria as seguinte representações possíveis:

Decimal	48
Hexadecimal	30h
Binário	00110000b

Obs: Números hexadecimais maiores que 7Fh devem ser precedidos por um zero, p. ex. 0F3h

A figura 3 mostra um exemplo de trecho de código em linguagem de montagem do NEANDER-X.

```
Simbólico       Comentários

ORG     0
LDA     X       ; o acumulador recebe o valor de X
ADD     Y       ; o acumulador é somado com Y
ADD     W       ; o acumulador é somado com W
STA     Z       ; o acumulador é copiado para Z
HLT             ; o processador para


ORG  150
Y:  DS    1     ; Endereço de Y definido como  150
W: DS     1     ; Endereço de W definido como 151
Z:  DS    1     ; Endereço de Z definido como  152
X  EQU  128     ; Endereço de X definido como  128

END
```

Figura 3 – Um programa para o NEANDER-X

5. O sistema simulador NeanderWin

Desde o início do projeto do simulador NEANDER-X nosso objetivo era facilitar ao máximo as atividades didáticas do professor e o apoio mais completo possível para as dificuldades comuns do aluno. Para isso foi criado um ambiente integrado para desenvolvimento, que executa em Windows e Linux, e que inclui:

- Editor de textos;
- Montador (assembler);
- Simulador da arquitetura;
- Visualizador da memória simulada;
- Ferramenta de apoio ao aprendizado de instruções;
- Utilitário para conversão de bases;
- Simulador de visor e painel de chaves;
- Gerador/carregador de imagem da memória simulada.

A Figura 4 mostra a aparência da tela principal do sistema NeanderWin. Na parte superior estão o menu geral de operação (Arquivo, Editar, etc...) e diversos botões usados em conjunto com o editor de textos, que seguem o estilo usual de programas Windows ou Linux.

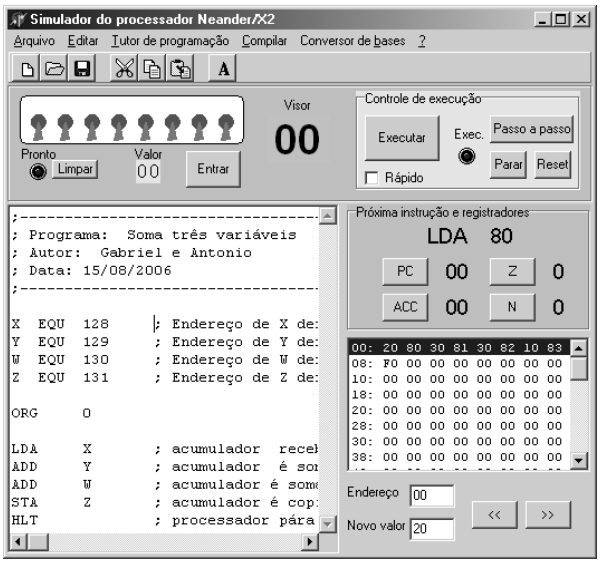

Figura 4 - Tela principal do NeanderWin

Logo abaixo, está o painel do simulador, em que são mostrados os dois dispositivos de entrada e saída (8 chaves e um visor em formato hexadecimal), e diversos botões para controle de execução.

Imediatamente abaixo, à esquerda, está o editor de textos, no qual o programa do aluno é digitado ou criado interativamente através de uma função para criação tutorada de programas, que será descrita com mais detalhes adiante.

À direita desta área se situam os verificadores dos registros e "flags" principais da CPU (ACC, PC, Zero e Negativo), e abaixo o visualizador da memória, com controles para alteração de conteúdo.

Uma vez criado o programa ele será compilado, o que provoca o aparecimento de uma janela "pop-up" com a listagem, num formato similar à maioria dos montadores profissionais, em que também são indicados nos eventuais erros de compilação. O

programa compilado se reflete na alteração da memória, que é imediatamente exibida no painel correspondente. Caso se deseje, é possível copiar o conteúdo desta janela para a área de transferência, para colar em algum editor (como o Word ou Bloco de Notas) possibilitando eventuais embelezamentos e impressão a posteriori.

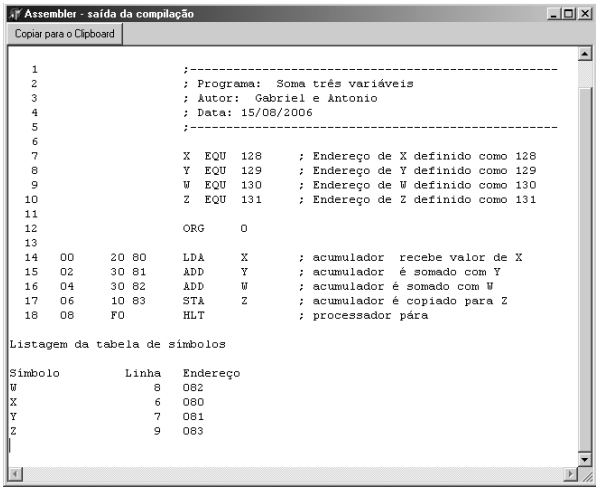

Figura 5 – Janela de listagem de compilação

O número de instruções do NEANDER-X é pequeno (apenas 16 instruções) mas apesar disso notamos que é importante tornar disponível uma "ajuda online" interativa, que é ativada pelo menu do programa, um sistema de entrada interativa de instruções e pseudo-instruções. A função de criação tutorada de programas, mostrado na Figura 6, faz com que o aluno cometa menos erros e a compreenda melhor o significado das instruções e produza um formato de instrução correto interativamente.

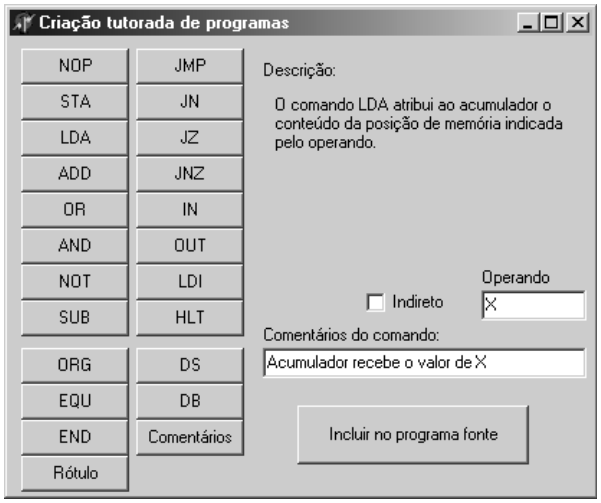

Figura 6 – Função de edição tutorada

Por último, notamos que para muitos estudantes o domínio das representações hexadecimal e binária, necessário para verificar e alterar a memória, só se dá depois de algum tempo. Mesmo sabendo que as calculadoras do Windows e do Linux exibem resultados em várias bases, incluímos um conversor de bases (mostrado na Figura 7), que é muito mais simples.

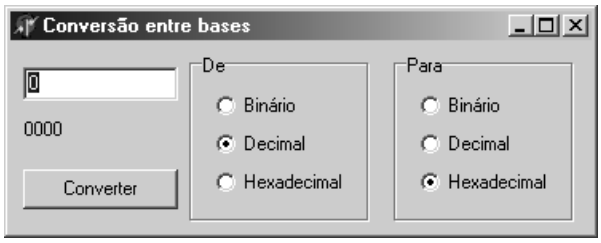

Figura 7 – Conversor de bases

6. Estratégias didáticas

O uso do Neander torna possível a introdução gradual dos conhecimentos de arquitetura de computadores, permitindo que os alunos façam diversas experiências práticas com o simulador. Busca-se nesse processo muito mais que apresentar conceitos teóricos e nomenclaturas, mas conduzir o aluno a uma visão abrangente e em pequena medida, crítica sobre a estrutura e o conjunto de instruções, que permita gerar curiosidade para estudos futuros mais aprofundados.

Descrevemos em seguida as possíveis fases do trecho do curso de arquitetura em que o NeanderWin é parte integrante:

a) Se inicia com uma breve explicação sobre significado do ACC, PC e flags, com exemplos em pequenos programas de 2 a 5 linhas.
b) Após o entendimento destes conceitos básicos, uma série de exercícios demonstram o uso das instruções em operações típicas de programação (movimentação de dados, loops, decisões, acesso a vetores), aplicados a pequenos algoritmos.
c) Terminada esta fase, os outros elementos arquiteturais (ALU, registradores, fluxos de dados, etc) são apresentados na forma de pequenas animações computacionais, geradas em Flash, sobre a Figura 1, mostrando o fluxo interno de dados de algumas instruções.
d) Concluindo, uma discussão final expande as idéias simples, consolidando uma nomenclatura mais geral de arquitetura de CPUs.

O item b (que é essencialmente de programação) traz em geral muito interesse nas turmas de informática, e é nele que deixamos que o potencial criativo dos alunos

floresça com mais vigor, deixando também o aluno com base para assimilar as fases c e d, muito mais pesadas do ponto de vista teórico.

7. Estudo de caso

O simulador NeanderWin foi utilizado como ferramenta de apoio para a disciplina de Arquitetura de Computadores do curso de Sistemas de Informação da Universidade Estácio de Sá desde 2004, com um público estimado de 1000 alunos, ao longo destes três anos.

Essa disciplina, assistida em geral no segundo período do curso de Análise de Sistemas, apresentava um quadro de dificuldade de aplicação para todos os professores, notadamente devido à apatia e falta de interesse dos alunos em compreender o funcionamento interno do computador e, em especial, do seu processador.

Neste curso noturno, muitos alunos eram provenientes do mercado, existindo também uma enorme disparidade de conhecimentos (desde pessoal de escritório até técnicos de manutenção, incluindo uns poucos programadores e analistas de sistemas formados na prática e à busca de um diploma).

Alguns alunos já traziam um conhecimento prévio sobre a arquitetura de computadores do tipo IBM/PC, focando o seu interesse em um nível superficial, principalmente sobre a maneira mais eficiente de interligar módulos e placas para incrementar o desempenho final do computador e quando apresentados a uma arquitetura teórica, era difícil estabelecer uma conexão entre sua vivência, muitas vezes com conceitos equivocados, e os conceitos teóricos apresentados sobre arquitetura de computadores.

Como tentativa de trazer o curso para uma visão de mercado, tentou-se usar no curso uma arquitetura comercial simples, explicada com poucos detalhes (8080), mas quando defrontados com a complexidade de detalhes intrínseca de um processadores real, isso era motivo de uma grande confusão, tanto para o professor, como para os alunos.

Após a introdução do uso do simulador NeanderWin, notamos em nossas turmas um aumento do interesse e participação dos alunos nas aulas em que o simulador era utilizado com uso do computador acoplado com recursos audiovisuais. O uso do simulador na sala de aula, através do uso de um computador conectado a um projetor de multimídia, permitiu, em curto espaço de tempo, o desenvolvimento e teste de soluções (em alguns casos, até mesmo mais de uma solução) para problemas simples.

Durante a aula problemas simples eram apresentados aos alunos, que produziam soluções no papel. Posteriormente uma solução "de professor" era apresentada e discutida na classe, criando grande interação entre os alunos. Em alguns casos, os próprios alunos sugeriam soluções diferentes, ou criticavam aspectos da solução apresentada. A partir da aplicação deste simulador, verificamos uma melhoria considerável na absorção dos conceitos abordados na disciplina e no interesse de estudar outros tópicos teóricos do curso que não eram cobertos pelo simulador.

Notamos também que o desempenho acadêmico (notas de prova), no mesmo curso e na mesma faculdade, dos alunos que faziam uso do simulador era superior aos alunos de turmas que não utilizavam este tipo de apoio didático. Infelizmente, não possuímos levantamentos estatísticos precisos a respeito, mas observamos que durante dois primeiros períodos a média das turmas que usaram o simulador ficou entre de 15 e 30 por cento acima das outras. Os formulários de avaliação, que são preenchidos pelos alunos ao fim dos períodos também mostraram avaliação superior para o quesito "uso de recursos pedagógicos". Estamos conscientes de que esses são dados passíveis de contestação, pois não estamos levando em consideração aspectos relativos ao desempenho pessoal dos professores.

8. Conduzindo o aluno da linguagem de alto nível à linguagem de máquina

Uma idéia que temos explorado em algumas turmas que têm melhor resposta pedagógica, consiste em fazer com que os alunos aprendam a aplicação semântica das instruções através da tradução manual de pequenos trechos de programas criados gerados numa linguagem muito simples (que denominamos MicroBasic), seguida se simulação no NeanderWin.[1]

Essa idéia leva o aluno a uma visão crítica sobre o uso das instruções, e produz como subproduto o entendimento empírico sobre o trabalho que um compilador tem que executar. Os pontos mais importantes da limitação deste dialeto são:

- Só variáveis inteiras, denominadas de A a Z;
- As operações aritméticas são limitadas a "+" e "-";
- Todas as linhas são numeradas;
- Comandos: INPUT, PRINT, LET, GOTO, IF, FOR, STOP.

[1] Como o MicroBasic é um dialeto muito simples de Basic, é também possível testar o algoritmo criado utilizando um interpretador qualquer desta linguagem, mas como os programas que criamos são sempre muito simples, não temos notícia de nenhum aluno que tenha feito tal execução.

1 REM programa exemplo	L10:
20 INPUT X	L20: IN 0 STA X
30 LET X = X + 1	L30: LDI 1 ADD X STA X
40 PRINT X	L40: LDA X OUT 0
50 STOP	L50: HLT

Figura 8 – Tradução manual de Microbasic

Mesmo em exemplos simples como o mostrado na Figura 8 é possível introduzir conceitos importantes como otimização de código e desempenho de algoritmos. Por exemplo, o comando LDA X pode ser eliminado sem prejuízo da execução.

Algumas traduções não são tão triviais, e estimulamos os alunos a buscarem soluções criativas, e validá-las no simulador. O trecho abaixo é uma solução criativa para a tradução do comando FOR.

40 FOR K = 1 TO 10	L40: LDI 1 STA K JMP L40B L40A: LDI 1 ADD K STA K LDI 11 SUB K JN L60B L40B:
....	....
60 NEXT K	L60: JMP L40A L60B:

Figura 9- Tradução manual mais sofisticada

Há diversos outros experimentos interessantes, como a manipulação de um "array" ou de ponteiros. Infelizmente o tempo necessário para explorar estes conceitos só nos deixa a alternativa de deixar esses exemplos mais sofisticados como exercícios para casa, o que nem sempre é muito efetivo.

9. Detalhes de implementação

O código original do sistema foi elaborado utilizando a linguagem Delphi, sendo que possui versões para Linux compiladas com o uso do Free Pascal com Lazarus. Foram escritas cerca de 3000 linhas de código em sua implementação.

O código foi projetado modularmente para que seja simples a inserção de novas instruções, novos modos de endereçamento e novos periféricos simulados. Busca-se incentivar estudantes mais "espertos" a criarem algumas instruções que são úteis para a criação de programação mais avançada, como instruções de multiplicação, divisão e deslocamento, chamada de rotinas e armazenamento de parâmetros.

O sistema faz uso intensivo dos objetos disponíveis no Delphi, que tem contrapartes em Lazarus e com pequenas diferenças também em Kylix. Assim, editor de textos, os objetos gráficos, e o tratamento de interação com o usuário usou sem necessidade de modificação os componentes padrões.

O montador é a parte mais complexa do sistema. É um montador clássico que trabalha em dois passos, sendo que na primeira varredura do fonte é feita uma tradução linha a linha do código, visando obter apenas o endereço dos rótulos, que são registrados numa tabela de símbolos implementada reusando objetos baseados em classes padrões de manipulação de listas. Os mnemônicos são organizados em uma outra tabela de acesso por busca binária. No segundo passo o montador relê o texto, e realiza a tradução final linha a linha, com os endereços à frente já resolvidos. Nesta fase o montador também copia o código gerado para a memória do simulador e produz a listagem da tradução.

Como os tamanhos dos programas fontes são minúsculos, não se buscou fazer uma grande otimização de seu código, mas mesmo assim, o programa leva cerca de um segundo para compilar 1000 linhas de código em linguagem de montagem num Pentium III.

O simulador é essencialmente um *loop* que interpreta as instruções através de um comando *case* de Pascal. Esse *loop* foi trabalhado cuidadosamente para permitir a execução opcionalmente em alta velocidade com pouca degradação da interatividade e do feedback detalhado na interface gráfica, possibilitando uma execução de depuração muito precisa, e produzindo uma execução "bastante realística" do código simulado.

Está disponível na Internet [11] uma distribuição binária para Windows com instalador automático. A versão para Linux está sendo neste momento preparada para distribuição.

10. Conclusões

Neste trabalho apresentamos o NEANDER-X, um processador de arquitetura muito simples, desenvolvido apenas para fins didáticos e um sistema de simulação com ferramentas integradas denominado Neanderwin.

A arquitetura e o conjunto de instruções do NEANDER-X, não permite comparação razoável com processadores comerciais, que são muito mais complexos que ele. Entretanto, seu uso didático é plenamente justificado não só pela sua simplicidade e rapidez de ensino, mas especialmente porque mesmo os processadores utilizados nas mais sofisticadas estações de trabalho são baseados nos mesmos conceitos elementares que são facilmente assimilados com o estudo do NEANDER-X.

O NeanderWin oferece uma interface de programação amigável, com a entrada do código em representação simbólica, com diversas facilidades para o programador, que tornam muito mais fácil o uso do processador NEANDER-X como ferramenta de ensino. Estão disponíveis versões tanto para o sistema operacional Windows e Linux. O código fonte está disponível gratuitamente mediante solicitação direta de aos autores.

Como todo projeto bem sucedido, muitas idéias têm aparecido para torná-lo mais abrangente e poderoso. Entre as principais idéias que provavelmente serão implementadas por nós ou por nossos parceiros podemos citar:

- Introdução de uma instrução para movimentação de dados entre PC e Acumulador (útil para possibilitar o uso de subrotinas);
- Introdução de operações de deslocamento de bits (útil para desenvolvimento de rotinas de multiplicação e divisão);
- Visualização “online” do fluxo de dados durante a execução das instruções;
- Animação dos fluxos de dados como apêndice do módulo de criação tutorada de programas;
- Construção de um Compilador “MicroBasic”, que gere como resultado a linguagem de montagem do Neander-X;
- Introdução de uma filosofia de "plug-ins", que permita a criação de periféricos externos;
- Extensão da arquitetura para 16 bits[2].

[2] Em particular já foi por nós construída uma versão "MIPS” usando o código deste sistema como base.

11. Bibliografia

[1] TANENBAUM, A. S. **Organização Estruturada de Computadores.** 3. Ed., Rio de Janeiro: Prentice-Hall, 1999.

[2] HENNESSY, J. L.; PATTERSON, D. A. **Arquitetura de Computadores: Uma Abordagem Quantitativa,** Tradução da Terceira Edição Americana, Rio de Janeiro: Campus, 2003.

[3] HENNESSY, J. L.; PATTERSON, D. A. **Organização e Projeto de Computadores:** A interface Hardware/Software**.** Rio de Janeiro: Elsevier, 2005.

[4] MURDOCCA, M. J.; HEURING, V. P. **Introdução à Arquitetura de Computadores**. Rio de Janeiro: Campus, 2001.

[5] YURCIK, W.; WOLFFE G; HOLLIDAY, M. A Survey of Simulators Used in Computer Organization / Architecture Courses, In: Summer Computer Simulation Conference, 2001, Orlando FL. USA. **Proceedings of the 2001 Summer Computer Simulation Conference**, July 2001. p. 524-529

[6] GRÜNBACHER, H. **WinDLX** Disponível em: http://www.soc.tuwien.ac.at/intern/RA/ Acesso em 26 Set. 2006

[7] SCOTT, M. **WinMIPS64** Disponível em: http://www.computing.dcu.ie/~mike/winmips64.html Acesso em 26 Set. 2006

[8] WEBER, R. F. **Fundamentos de Arquitetura de Computadores.** 2. Ed. Porto Alegre: Instituto de Informática da UFRGS: Sagra Luzzatto, 2001.

[9] BORGES, J. A. S. ; FALLER, N. O CP/M No Unix - Uma Experiência de Emulação. In: Congresso Regional da SUCESU, 1987, Brasília. **Anais do Congresso Regional da SUCESU**, Brasília: SUCESU, 1987.

[10] BORGES, J.A E SCHMITZ, E.A. **Projeto de Circuitos Integrados.** Rio de Janeiro: LTC, 1990.

[11] SILVA, G. P.; BORGES, J. A. S. **O Simulador Neanderwin** Disponível em: http://equipe.nce.ufrj.br/gabriel/neanderwin Acesso em 26 Set. 2006