Relato de Experiência Interdisciplinar Usando MIPS

Sílvio Roberto Fernandes
Departamento de Computação
Universidade Federal Rural do Semi Árido, UFERSA
Mossoró, Brasil
silvio@ufersa.edu.br

Ivan Saraiva Silva
Departamento de Computação
Universidade Federal do Piauí, UFPI
Teresina, Brasil
ivan@ufpi.edu.br

***Resumo* — O desafio de despertar o interesse dos alunos de ciência da computação para disciplinas voltadas para o *hardware* tem proporcionado criação de metodologias e experimentos de ensino. Comumente essas soluções são isoladas para cada disciplina. Assim, este artigo apresenta um relato de um conjunto de atividades práticas nas disciplinas de Organização e Arquitetura de Computadores e Sistemas Operacionais com base no MIPS e suas ferramentas, principalmente o MARS. A aplicação dessas práticas tem se mostrado efetiva na taxa de sucesso das disciplinas e sua utilização também tem sido aprovada pelos alunos.**

Palavras-chaves* — *Arquitetura de Computadores, Sistemas Operacionais, MIPS, MARS, Interdisciplinar

I. INTRODUÇÃO

Os cursos de graduação em Ciência da Computação, em sua grande maioria, apresentam uma grande abrangência nas diversas subáreas dentro da computação, desde o *hardware* até o *software*. Provavelmente pelo maior enfoque em *software*, há um certo desinteresse dos alunos nas disciplinas mais voltadas para os aspectos físicos ou funcionais das máquinas. Outro fator que pode afetar neste mesmo sentido é a falta de integração entre as disciplinas, o que tornam obscuros a importância e o propósito delas na formação do profissional em computação. Esses, entre outros motivos, tornam essas disciplinas responsáveis por grande parte da retenção e evasão dos alunos, considerando as disciplinas específicas de computação. É importante que professores de disciplinas que chamaremos de "disciplinas de hardware ou software básico" motivem os alunos mostrando que entender os seus conceitos torna-os melhores programadores e permite que tirem o melhor proveito das máquinas.

Neste sentido, é importante o desenvolvimento e aplicação de metodologias de integração interdisciplinar que ao mesmo tempo estimulem o interesse dos alunos e promovam o raciocínio e a criatividade. Do mesmo modo, é importante alinhar teoria e prática no processo de ensino e aprendizagem. Na literatura são encontradas diversas metodologias que tem mostrado bons resultados a respeito da redução da evasão ou melhoria das notas, entretanto elas são, em sua maioria, específicas para uma disciplina, sem integração interdisciplinar.

Portanto, este artigo relata uma experiência interdisciplinar, por meio de atividades práticas envolvendo o processador MIPS nas disciplinas Arquitetura de Computadores (OAC), Sistemas Operacionais (SO), promovendo assim uma integração entre elas. O objetivo desse relato é inspirar professores e pesquisadores a desenvolverem atividades que integrem disciplinas e despertem o interesse dos alunos para que possam diminuir evasão, melhorar o aprendizado e formar melhores profissionais em computação. Conceitualmente, não foi desenvolvida uma nova metodologia, entretanto, ao longo do texto será usada a palavra "metodologia" para referenciar o conjunto de práticas usadas nas disciplinas.

O artigo está organizado da seguinte forma: a seção II apresenta uma contextualização para este artigo, destacando alguns trabalhos relacionados, além do contexto local onde foram realizas a experiências deste artigo, bem como o ambiente para o desenvolvimento delas; a seção III apresenta o conjunto de atividades práticas nas disciplinas de OAC e SO; a seção IV realiza uma avaliação da metodologia proposta; e a seção V a conclusão seguida pelas referências.

II. CONTEXTUALIZAÇÃO

A. *Trabalhos Relacionados*

A metodologia de aprendizado baseado em problemas (PBL – *Problem Based Learning*) foi proposta inicialmente para medicina mas tem sido adaptada para diversas áreas, como computação. Exemplos específicos para o ensino de OAC encontram-se em [1] e [2]. Esses trabalhos apresentam prós e contras da metodologia, mas concluem que há um saldo positivo em sua aplicação em OAC devido sua abrangência do conhecimento e exigir soluções criativas.

Outra abordagem é o ensino baseado em projetos, como a apresentada em [3] juntamente com a elaboração de um material didático. Os resultados quantitativos da metodologia foram obtidos por meio de enquetes com a opinião dos alunos em diversos semestres aplicada por diferentes professores. Os autores consideram como vantagens dessa metodologia: os alunos mais motivados para projetar e tomar decisões e, para os professores, mais dinâmica para novas situações-problemas a cada semestre. Como desvantagens: dedicação intensa do professor e a necessidade de ferramenta de ensino a distância bem instalada, funcional e sempre disponível.

Em [4] são propostas duas novas metodologias para o ensino de OAC: baseado em analogia (AEM – *Analogy-Example Method*) e orientada a aplicação (ADEM –

Application-Driven Extension Method). A avaliação da sua proposta foi realizada comparando as notas dos alunos em um semestre sem e outro com as metodologias. A melhora nas notas os fez concluir que as metodologias tornam o aprendizado mais efetivo.

Um dos principais aliados da ligação entre teoria e prática das disciplinas de *hardware* ou software básico são os simuladores, os quais podem ser encontrados em diversos trabalhos que relacionam o uso de uma metodologia com algum deles. Em [5] foi realizada uma análise comparativa entre simuladores de sistemas completos para o ensino de OAC. Essa análise levou em consideração as características dos simuladores (licença, tipos de arquitetura simuláveis, granularidade de parametrização, modos de simulação, suporte a *debugging*, suporte a *tracing* e documentação) e uma avaliação em sala de aula pelos alunos por meio de um questionário com onze perguntas de múltipla escolha. Os autores concluíram que nenhum dos simuladores possui todas as características didáticas desejáveis. Sob a ótica dos alunos, por meio do questionário, os simuladores são aprovados como ferramenta de ensino, mas sua utilização é dificultada possivelmente por documentação insuficiente.

Dentre os vários processadores utilizados em disciplinas de *hardware* ou software básico, podemos destacar o MIPS (*Microprocessor without Interlocked Pipe Stages*) [6], [7], que dentre suas características de alto desempenho e regularidade das instruções, é um modelo que ao mesmo tempo é usado em produtos comerciais como também didaticamente, e por isso, apresenta diversos simuladores. Em [8] é feita uma análise comparativa de diversos simuladores do processador MIPS encontrados na literatura (WebMIPS, DIMIPSS, MARS, WinMIPS64, MipsIt) e o proposto pelos autores (uma extensão do Ptolemy). Nessa análise, verificou-se que o Ptolomy pode ser usado nas disciplinas de OAC básica e avançada e dar suporte as principais características analisadas. Na avaliação de usabilidade se destacou principalmente em *feedback*, flexibilidade e proteção contra erros.

Considerando a disciplina de SO, o artigo [9] é intitulado como um *framework* customizável para ensino dessa disciplina em diferentes níveis que combina aspectos teóricos e práticos. A metodologia proposta pelos autores é apresentada em dois livros (um experimental e um teórico), contudo o artigo não apresenta o *framework* e sim os livros gerados a partir dele. Em [10] é proposta a utilização de jogos para melhorar o aprendizado em SO. Inicialmente são propostos os jogos "palavras cruzadas" e o jogo perguntas e repostas "Jeopardy!" para qualquer disciplina. Em seguida são apresentados os dois jogos específicos para SO, propostos pelo artigo: "batalha de *threads*", inspirado em batalha naval; e "jogo da transição de estado dos processos". Os alunos aprovaram essa metodologia indicando que aprenderam mais rapidamente as diferenças e vantagens de *threads* e processos. Já o artigo [11] propõe o uso de um núcleo virtual de um Linux, implementado pelos autores, que pode ser modificado, depurado e reiniciado sem afetar a instalação no computador nem outros usuários. Além disso, também permite o desenvolvimento de atividades de ensino à distância. Em [12] é proposta uma abordagem baseada em problema, por meio de dois estudos de caso: escalonamento de processos e o problema de *deadlock*. A avaliação dessa metodologia realizada por professores e alunos concluiu que sua aplicação melhorou as habilidades de aprendizado e senso crítico dos alunos, os quais esperam que a metodologia continue sendo aplicada.

Levando em consideração propostas interdisciplinares, podemos citar o cMIPS [13], uma implementação VHDL do processador MIPS e alguns periféricos que podem ser usados nas disciplinas Circuitos Digitais, Microprocessadores, Arquitetura, Software Básico e Sistemas Operacionais. Contudo, quando esse artigo foi publicado essa plataforma não tinha sido utilizado ainda nas disciplinas. Em [14] é apresentada uma metodologia de desenvolvimento de circuitos digitais, usando a construção de uma ULA como estudo de caso. De acordo com os autores a metodologia foi experimentada, com êxito em disciplinas como Arquitetura de Computadores e Organização de Computadores para o curso de Ciências da Computação, Circuitos Digitais, Microprocessadores e Microcontroladores II e Eletrônica Digital e Ao Projeto VLSI II para o curso de Engenharia Elétrica. Os autores não apresentam uma análise do uso da metodologia. Outra proposta interdisciplinar é a família de processadores BIP (*Basic Instruction-set Processor*) [15] e seu ambiente BIPIDE [16], que formam uma plataforma. Os autores sugerem como essa plataforma pode ser usada nas disciplinas Algoritmos e Programação, Circuitos Digitais, Compiladores e OAC, propondo materiais para a última disciplina. Em [17] é proposta uma prática de ensino de circuitos lógicos que levam a construção de um processador básico. A metodologia envolve o uso da ferramenta KTechLab [18], uma adaptação do método socrático por meio do jogo *Cisco Binary Game* [19], modificação de rotinas em linguagem C para estudo de aritmética binária de inteiros e flutuantes, atividades com circuitos físicos e *protoboard* para estudo de portas lógicas e para o estudo de um processador básico de 4 bits com 4 registradores e display de 7 segmentos. A metodologia foi avaliada pelos alunos graduando de 1 a 5 as questões de facilidade de aprendizado e motivação, e obtiveram respectivamente média de 4,7 e 4,6, atingindo o objetivo proposto.

B. Contexto Local

A metodologia proposta neste artigo foi utilizada em turmas de Ciência da Computação da UFERSA (Universidade Federal Rural do Semi Árido) no turno noturno, que é o horário de funcionamento do curso. O turno do curso já é um desafio para manter o estímulo dos alunos que muitas vezes chegam cansados de um dia de trabalho e não dispõem de muito tempo para atividades extraclasse. Esse curso tem entrada semestral, logo, todas as disciplinas obrigatórias são oferecidas todos os semestres, enquanto que as opcionais dependem de uma demanda específica ou distribuição de carga horária. Na grade curricular desse curso, a disciplina Circuitos Lógicos é pré-requisito de Organização e Arquitetura de Computadores (OAC), que por sua vez, é pré-requisito de Sistemas Operacionais (SO) e Sistemas Embarcados (SE). Essa última é uma disciplina opcional e as demais são obrigatórias. Também vale destacar que o autor deste artigo é o professor das disciplinas citadas, exceto Circuitos Lógicos. Todas as disciplinas são divididas em 3 unidades, tendo cada unidade

uma nota, as quais são usadas para calcular uma média ponderada. Aqueles que atingem a média mínima estão aprovados e para os demais há uma quarta avaliação (recuperação) e o cálculo de uma nova média.

Observou-se ao longo dos semestres uma grande retenção na disciplina OAC e consequentemente no atraso nas outras que a usam como pré-requisito. Dessa forma, tem se buscado constantemente por estratégias e metodologias que despertem o interesse dos alunos, que melhorem o ensino/aprendizagem e, consequentemente, diminuam a evasão e a retenção dos mesmos. É importante notar que estratégias bem-sucedidas em uma turma muitas vezes não tem o mesmo efeito em outras, já que cada turma possui um contexto único. Um levantamento relativo à aprovações, reprovações e desistências na disciplina de OAC foi realizado entre os semestres 2010.1 e 2014.1. Na Fig. 1 observamos uma predominância de uma taxa de reprovação de no mínimo 50%.

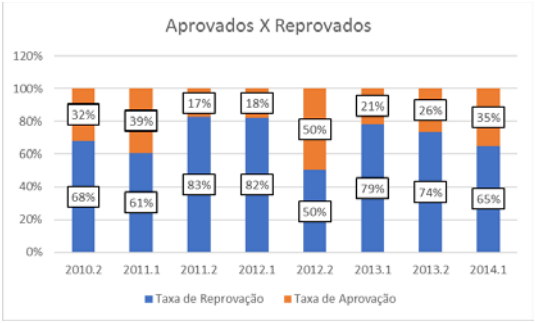

Fig. 1. Taxa de aprovados versos taxa de reprovados em OAC

A Fig. 2 apresenta os motivos das reprovações (por nota ou por falta). A média geral das reprovações por falta (46%) mesmo menor que por nota (54%) é preocupante, o que significa quase metade dos reprovados na verdade desistiram da disciplina antes do seu fim.

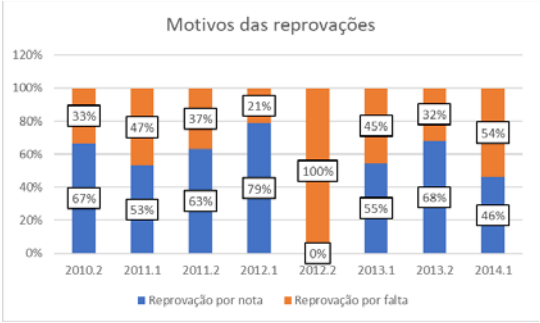

Fig. 2. Motivos das reprovações em OAC

Desde 2012 passou-se a se utilizar o livro do D. Patterson [20], [25] como literatura básica, que apresenta o processador MIPS, e consequentemente o uso de simuladores para desenvolvimento das atividades com *assembly* desse processador. Em 2014 foi introduzida a linguagem VHDL para o desenvolvimento de processador MIPS divido em três partes, ao longo do semestre, como parte obrigatória para a nota. As atividades eram incrementais até a construção completa de um processador MIPS simplificado. Mas a dificuldade com a linguagem e com um novo paradigma aumentava o desestímulo, que pôde ser observado com a diminuição da quantidade de trabalhos entregues na sequência das atividades.

Em 2014 também foi realizado um questionário com a turma a respeito da metodologia que foi empregada naquele momento. E o resultado desse questionário é resumido na Tabela 1. A maior parte das atividades adotadas foram aprovadas pelos alunos.

TABELA 1. QUESTIONÁRIO APLICADO EM 2014

PERGUNTA	% RESPOSTAS
Você acha que a disciplina é muito teórica?	81,8% sim
Uso de simuladores melhora ou atrapalha o entendimento dos conceitos?	100% melhora
Você acha que a disciplina seria melhor se fosse ministrada em laboratório?	90,9% sim
Você acha que a disciplina poderia ter mais atividades práticas SEM diminuir a teoria?	72,7% sim
Você acha que a disciplina poderia ter mais atividades práticas DIMINUINDO a teoria?	54,5% não
Você acha que usar linguagens de descrição de hardware (como VHDL) durante a maior parte do curso poderia ajudar no entendimento dos conceitos?	54,5 não
Você acha que deveriam haver trabalhos de implementação OBRIGATÓRIOS como parte da nota?	63,6% não
Você acha que deveriam haver trabalhos de implementação OPTATIVOS como nota extra?	100% sim
Você acha que os trabalhos de implementação sugeridos nessa disciplina em 2014.2 foram interessantes?	63,6% sim

Com a análise da participação dos alunos nas atividades de implementação em VHDL e o *feedback* dos alunos, foi observado que a obrigatoriedade dessa implementação prejudicava mais que ajudava, aumentando as reprovações e desestímulos, uma vez que os alunos não tinham maturidade suficiente para desenvolvê-las. Logo, essa avaliação guiou a metodologia proposta que será apresentada na próxima seção.

Na disciplina de SO, mesmo não havendo altas taxas de reprovações e retenções, ainda se almejava a utilização de atividades práticas. A princípio foi utilizado o sistema operacional real MINIX [21] e o livro texto do seu autor [22]. Essa metodologia alia bem a teoria e prática, uma vez que o autor é um dos mais utilizado em cursos de SO. Contudo, a versão do sistema documentada no livro (versão 3.1) possui poucas e defasadas ferramentas, o que dificultou muito o desenvolvimento das atividades. O sistema evoluiu, como se vê no site oficial, mas nenhuma edição nova do livro foi lançada. Desse modo, o uso do MINIX foi abandonado e a familiaridade dos alunos com o processador MIPS e seu *assembly* vistos na disciplina de OAC foi aproveitada, uma vez que o ambiente de simulação possibilita a expansão das ferramentas para o desenvolvimento de atividades práticas também na disciplina de SO.

Logo, este artigo propõe uma metodologia de desenvolvimento de atividades práticas para as disciplinas de OAC e SO com as ferramentas e implementações apresentadas a seguir.

C. Ambiente Utilizado nas Atividades

O ambiente utilizado para realizar as atividades da metodologia proposta está relacionado ao processador MIPS [6], [7]. Há diversos modelos desse processador, mas o levado em consideração foi o apresentado em [20]. Essa versão possui palavras de 32 bits, 32 dois registradores, sendo 18 disponíveis para o programador e memória de 2^{30} palavras. A ISA (*Instruction Set Architecture*) apresentada é formada por um conjunto reduzido de instruções totais do processador, entre as quais, lógicas, aritméticas, desvio condicional e incondicional, acesso a memória e chamada e retorno de função. Essas instruções são divididas em 3 tipos: R, I e J, sendo todas elas com mesmo tamanho (32 bits) e código de operação nos 6 bits mais significativos.

Para aplicação da metodologia proposta foi utilizado o simulador MARS [7] e uma implementação parcial em VHDL do processador MIPS apresentado em [23].

O MARS (*MIPS Assembler and Runtime Simulator*) é um IDE (*Integrated Development Environment*) implementado em Java que disponibiliza um editor de texto, um montador MIPS e um simulador. O montador faz verificação de sintaxe do código *assembly*, oferece um conjunto de chamadas de sistema que simula diversos serviços de sistema operacional durante a execução dos programas e pseudoinstruções, as quais são substituídas pelas instruções correspondentes durante o processo de montagem, aumentando a abstração do programador. A simulação apresenta informações dos segmentos de memória (código e dados), mostrando os endereços e os respectivos conteúdos, todos os registradores e um coprocessador de ponto flutuante. As informações na memória e registradores são atualizadas em tempo de execução destacando de forma colorida para percepção do usuário. A simulação também pode ser controlada por parte do usuário que escolhe em executar todas as instruções de uma vez ou quantas instruções por segundo ou no modo passo-a-passo que acontece apenas ao clique do usuário, sendo possível também pausar, retroceder e parar a execução. Integrado ao simulador há um conjunto de ferramentas (*tools*) que podem ser conectadas durante a simulação para observar os mecanismos internos do simulador e fornecer ao usuário mais informações ou opções de interação do processador com elementos externos. O MARS ainda oferece, em sua implementação, classes abstratas que podem ser implementadas ou classes que podem ser estendidas para criação de novas chamadas de sistema, novas ferramentas, novas pseudointruções ou novas instruções. O tutorial do MARS [24] apresenta como estender todas essas funcionalidades.

No livro de Harris e Harris [23], a arquitetura MIPS é apresentada após a conceituação de circuitos digitais e uma introdução a linguagens de descrição de *hardware* (VHDL e SystemVerilog). A apresentação da organização do MIPS é feita de maneira incremental juntamente com implementações equivalentes em VHDL e SystemVerilog, até ser exibida uma versão final funcional em ambas as linguagens. Essa versão inclui as principais instruções de cada um dos 3 tipos de instrução (as mesmas apresentadas em [20]) e utiliza arquivos de texto para simulação das memórias. Para o desenvolvimento das atividades foi utilizado o código VHDL com algumas pequenas modificações para uso com o kit de prototipagem DE2-115 da Altera [26].

III. Metodologia Proposta

Nesta seção é apresentada a metodologia usada no desenvolvimento das atividades práticas nas disciplinas OAC e SO, todas voltadas para o processador MIPS e suas ferramentas. As atividades de ambas as disciplinas podem ser acessadas na página web do autor[1], na seção "Textos Didáticos".

A. Atividades em Organização e Arquitetura de Computadores

As atividades práticas propostas para cada unidade dessa disciplina são opcionais e correspondem a nota extra apenas na unidade correspondente.

O conteúdo da primeira unidade da disciplina corresponde a: cálculo de desempenho dos programas, relacionado ao tempo de execução; programação *assembly* do MIPS; aritmética binária.

A parte de cálculo do desempenho é apresentada teoricamente, juntamente com exemplos e exercícios propostos.

Antes de conhecer o processador MIPS propriamente dito, os alunos são introduzidos ao modelo de programação *assembly*. Nesta parte são usados como literatura básica o livro do Patterson [20] e como complementar o [27] que é voltado principalmente para o *assembly* do MIPS. Nesse ponto os alunos também são introduzidos ao ambiente MARS [7], onde passam a desenvolver as atividades práticas dessa unidade. Aqui também é apresentada a ferramenta adicional do MARS (*tool*) que conta os tipos de instruções, da qual pode-se extrair informações para calcular o desempenho de soluções diferentes para os mesmos problemas.

Na parte de aritmética binária, é feita uma revisão de representação de inteiros com e sem sinal e soma e subtração binária. Em seguida são adicionados algoritmos de multiplicação e divisão binária e representação de ponto flutuante. Por fim são apresentadas as instruções do MIPS correspondentes a toda essa aritmética e simulação no MARS. Nos exercícios é vista a diferença entre inteiros com e sem sinal e como o *overflow* pode ser detectado pelo MARS.

Como atividade prática dessa unidade, é proposta a construção de uma biblioteca de *string* em *assembly* (com as funções *strlen*, *strcmp*, *strcat*, *strncat*, *strncpy*) usando a convenção MIPS para funções. Para testar, deve ser implementado também um código *main* com um menu de escolha pelo usuário para cada função.

A segunda unidade da disciplina é iniciada com apresentação de um código *assembly* muito simples o qual é executado em um MIPS sintetizado em FPGA no kit Altera DE2-115 [26]. O projeto sintetizado utiliza uma interface de controle de execução do programa, pelo usuário, com exibição

[1] https://sigaa.ufersa.edu.br/sigaa/public/docente/producao.jsf?siape=1566120

das informações do processador de modo semelhante ao MARS.

A apresentação do FPGA objetiva motivar os alunos a querer entender como aquela placa está executando o programa exemplo. Isso facilita realizar uma introdução a linguagens de descrição de *hardware*, usando VHDL. O código de exemplo é gerado em formato .mif (*memory initialization format*) e compilado junto com o projeto VHDL. Esse código no formato .mif é gerado pelo próprio MARS na opção "*dump memory contents to file*", entretanto foi necessário descomentar o trecho de código que faz isso e recompilar o MARS.

Em seguida, o projeto do caminho de dados do processador MIPS ciclo único vai sendo construído para cada classe de instrução de forma incremental, até chegar ao modelo completo onde é apresentado seu código VHDL adaptado de [23]. De forma análoga é apresentado o controle do processador e o respectivo código VHDL. Durante a apresentação do caminho de dados também é usada a ferramenta MIPS X-Ray [28] presente no MARS a partir da versão 4.5.

No final dessa unidade, é apresentado o projeto do MIPS multiciclo do Patterson [20] como uma forma teórica de melhorar o desempenho.

A atividade prática da segunda unidade corresponde a utilizar o código VHDL apresentado, para incluir cinco novas instruções (bne, sll, srl, jr e jal). Para isso é necessário modificar o projeto do caminho de dados e controle antes de efetivamente alterar o VHDL. Para testar o aluno deve implementar um código VHDL de *benchmark* e sua simulação.

Na terceira unidade é apresentado o projeto da versão do MIPS com pipeline para melhorar o desempenho de execução. Para ilustrar o funcionamento do pipeline é utilizada a ferramenta MIPSFPGA [29]. Nesta unidade também é abordada a hierarquia de memória e usadas as ferramentas do MARS para visualização de memória cache e de memória virtual.

A atividade prática da terceira corresponde a modificação do código VHDL de ciclo único para uma versão com pipeline (sem considerar os *hazards*) e também devem apresentar um *bechmark* e simulação.

B. Atividades em Sistemas Operacionais

Sempre houve dificuldade em desenvolver atividades práticas em SO em tempo viável e sem a necessidade instalação de várias ferramentas ou de um *background* prévio do sistema operacional a ser estudado. Dessa forma, foi aproveitado o conhecimento dos detalhes do processador MIPS e do ambiente MARS adquiridos na disciplina OAC para o desenvolvimento de atividades práticas focadas nos objetivos de SO. As propostas de atividades práticas que serão apresentadas já foram usadas como as avaliações de forma parcial, mas também de forma total de cada unidade em SO.

A primeira unidade dessa disciplina é composta por uma revisão de OAC, processos e *threads*, comunicação entre processos (IPC) e problemas clássicos de IPC. Durante a revisão de OAC é usado o MIPS como exemplo, assim como uma revisão do MARS, que ajuda a nivelar alunos que vem de outras instituições. Adicionalmente é realizado um tutorial de como estender suas funcionalidades por meio de novas *syscalls* e novas ferramentas, de acordo com [24]. Como o MARS é disponibilizado em um formato .jar, é possível acessar seu código, realizar as modificações e recompilá-lo.

Assim, nessa unidade, a atividade prática proposta consiste no desenvolvimento de 3 *syscalls* relativas ao gerenciamento de processos. Nessa atividade não são consideradas interrupções, de modo que não há escalonamento preemptivo. Logo, o aluno deve criar em java as classes: PCB (*Process Control Block*), Tabela de Processos e Escalonador para integrá-las ao MARS. A PCB guarda as informações de um processo (identificador, conteúdo dos registradores e estado do processo) e permiti que seja estendida para os demais gerenciadores. A Tabela de Processos permite a criação e remoção de PCBs, incluir e retirar os PCBs em uma lista de processos no estado de pronto e indicar qual está em execução. A classe Escalonador implementa o algoritmo de escolha, entre os processos prontos, de qual processo vai executar na CPU. Essa classe pode implementar diversos algoritmos, o qual pode ser escolhido em tempo de projeto ou de execução. Nesse primeiro trabalho, os alunos implementam apenas o algoritmo de fila por ser mais simples. Todas essas classes devem fazer parte de um pacote "gerenciador de processos". As *syscalls* também são classes java que conseguem interagir com os componentes do simulador que são "observáveis".

O aluno também deve criar 3 syscalls: *Fork*, *ProcessChange* e *ProcessTerminate*. A *syscall Fork* recebe como parâmetro o endereço inicial do programa (que será marcado por um *label* comum no código *assembly*) e é responsável pela criação de um PCB para o processo, o qual é incluído na tabela de processos no estado de pronto. A *syscall ProcessChange* indica que o processo está abrindo mão voluntariamente da CPU. *ProcessChange* primeiramente chama o algoritmo de escalonamento para indicar qual processo deve executar. Em seguida, essa *syscall* deve verificar se há algum processo atualmente em execução, em caso afirmativo salvar o contexto dele (valores dos registradores físicos do processador) para sua respectiva PCB, e carregar os valores do PCB do novo processo escolhido para o processador. Caso a CPU estivesse ociosa apenas o carregamento a partir da PCB é necessário. Se um processo deixa a CPU, ele é colocado no estado "pronto" e quando ganha a CPU é colocado no estado "executando". A terceira *syscall*, *ProcessTerminate*, indica que o processo está sendo encerrado de modo que sua PCB é removida da tabela de processos e consequentemente não é mais escolhido pelo escalonador para executar.

A Fig. 3 apresenta um código *assembly* para testar o gerenciador de processos. A definição das *sycalls* (passagem de parâmetros, o número e a instrução *syscall*) estão definidas no arquivo "macros.asm" não apresentado nessa figura. Nesse exemplo são criados 3 processos, onde os 2 primeiros irão executar um laço com um número limitado de iterações, enquanto que o último executará um laço infinito. Em seguida, o escalonador é chamado (por meio de *ProcessChange*) para executar um dos processos. Os processos (Programa 1 e Programa 2) propositalmente manipulam os mesmos registradores ($s1 e $s2) e chamam o escalonador a cada

iteração do laço. Isso permite os alunos verificarem o isolamento entre os processos e que a execução de um não interfere no outro.

```
.include "macros.asm"

.data
.text
        #criação dos processos
        fork(Programa1)
        fork(Programa2)
        fork(Idle)
        #escolnando o primeiro processo
        ProcessChange

Idle:
        loop:
                ProcessChange
                j loop

Programa1:
                addi $s1, $zero, 1 # valor inicial do contador
                addi $s2, $zero, 10 # valor limite do contador
        loop1:  addi $s1, $s1, 1
                beq $s1, $s2, fim1
                ProcessChange
                j loop1
        fim1:   ProcessTerminate

Programa2:
                addi $s1, $zero, -1 # valor inicial do contador
                addi $s2, $zero, -10 # valor limite do contador
        loop2:  addi $s1, $s1, -1
                beq $s1, $s2, fim2
                ProcessChange
                j loop2
        fim2:   ProcessTerminate
```

Fig. 3. Código assembly para testar o gerenciador de processos

Para o MARS o código da Fig. 3 possui apenas um programa, o qual é alocado no mesmo espaço de memória, então, de fato, são threads de um mesmo processo, de modo que isso fica claro para os alunos.

A segunda unidade de SO corresponde ao estudo dos algoritmos de escalonamento e gerenciamento de memória. A atividade proposta está relacionada com o último assunto.

Inicialmente os alunos devem modificar a classe PCB e a *syscall Fork* para incluir os endereços dos segmentos de dados, código e pilha, bem como registradores de limite máximo e mínimo do espaço de endereçamento do processo. Qualquer acesso indevido no espaço de endereçamento de outro processo deve interromper a execução desse processo e o usuário deve ser alertado. Além disso, os alunos também devem construir uma ferramenta (*tool*) do MARS para simular a técnica de memória virtual. Para isso o gerenciador também deverá armazenar os atributos globais para todos os processos: tamanho de bloco de alocação de memória (tamanho da página virtual) em quantidade de instruções do MIPS, quantidade máxima de blocos de alocação por processo e configuração do tipo de algoritmo de substituição de páginas. Todos esses atributos devem ser configurados pelo usuário por meio da *tool* criada. Esse gerenciador também deve manter uma tabela de páginas virtuais, com um conjunto de entradas na tabela para cada processo no momento que ele é criado por um *Fork*. Nas entradas da tabela são armazenados os atributos: bits referenciada e modificada, bits de proteção (R, W, X), bits presenta/ausente e número da moldura da página na memória física.

A simulação da memória virtual faz as seguintes considerações: quando os processos são criados eles ainda não estão alocados na memória; e a memória do MARS para a tool criada funciona como o disco. A medida que os processos vão sendo executados, o gerenciador de memória vai traduzindo os endereços virtuais em endereços físicos e atualizando na tabela de memória virtual, contabilizando *miss* e *hits* e invocando o algoritmo de substituição de páginas quando necessário. Todas essas informações são exibidas para o usuário por meio da *tool* desenvolvida na atividade. Também são configurados os tempos de acesso (dado em quantidade de instruções executadas) a memória e ao "disco" para realizar uma comparação do desempenho dos algoritmos de substituição na execução do mesmo teste. O teste utilizado pode ser o mesmo do da atividade da primeira unidade ou alguma variante. Na Fig. 4 é mostrada um exemplo de uma *tool* desenvolvida pelos alunos nesta atividade.

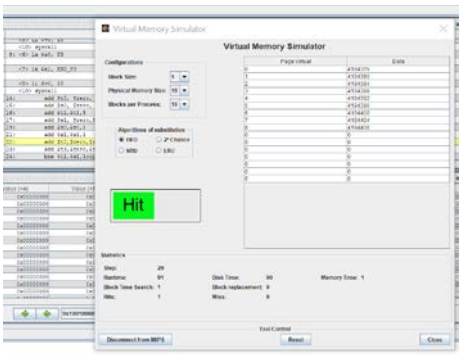

Fig. 4. *Tool* de memória virtual para o MARS

A unidade três de SO corresponde ao sistema de arquivos e ao gerenciamento de entrada e saída. A atividade prática proposta aborda o último assunto. Assim, os alunos devem desenvolver uma nova *tool* que simule um *timer*, o qual gera interrupção para desenvolvimento de um escalonador preemptivo. Essa *tool* deve permitir o usuário configurar o tempo de geração de interrupções (medido em número de instruções executadas). Quando o *timer* atinge seu limite, uma interrupção é gerada para o processador e a contagem é zerada para reiniciar. A Fig. 5 apresenta um exemplo da *tool* desenvolvida pelos alunos para essa atividade.

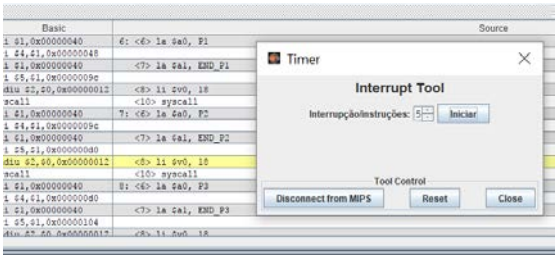

Fig. 5. *Tool* para geração de interrupção no tempo

Essa interrupção substitui o trabalho da *syscall ProcessChange* (proposto na primeira unidade), logo essa *syscall* não deverá ser mais utilizada. Adicionalmente, nesse trabalho também deverá ser modificada a *sycall PrintString*, original do MARS, que antes imprimia a string imediatamente passada por parâmetro, agora deverá esperar um tempo (dado em quantidade de instruções executadas, configurada previamente), simulando a demora da E/S. Dessa forma, quando um processo chamar *PrintString*, este processo é colocado no estado "Bloqueado" e o escalonador é invocado para escolher outro processo "Pronto". Enquanto o processo estiver "Bloqueado" não poderá ser escolhido pelo escalonador para executar. Quando o tempo da E/S terminar, a *string* passada para *syscall PrintString* é impressa e o processo "Bloqueado" deve ser colocado em "Pronto" para ser tornar elegível pelo escalonador novamente. Para testar as funcionalidades dessa atividade deve se implementar um código *assembly* que cria vários processos, todos em laços

infinitos, alguns chamando a *syscall PrintString* e outros não, sem uso de *ProcessChange*.

IV. Avaliação da Metodologia

Para avaliar a metodologia proposta utilizamos quatro critérios: (1) dados relacionados a desistência das turmas (antes e depois da metodologia); (2) dados relacionados a aprovação; (3) análise das notas; (4) e avaliação dos alunos por meio de um questionário sobre a metodologia.

Um dos principais objetivos da aplicação dessa metodologia era aumentar o estímulo dos alunos com relação às disciplinas, e possíveis consequências seriam diminuir a evasão e a taxa de reprovação. Dessa forma, a Fig. 6 exibe o estudo realizado com os dados sobre desistência (evasão) nas turmas de OAC durante 13 semestres (de 2010.2 a 2016.2).

Fig. 6. Dados sobre desistência em OAC antes e depois da metodologia

Essa figura mostra gráficos em barras correspondentes aos percentuais de desistências dos semestres antes da aplicação da metodologia (2010.2 a 2013.2) e depois da aplicação (2014.1 a 2016.2). Na mesma figura, é apresentada uma curva (em cinza) com os valores absolutos dos desistentes para cada semestre, e dessa forma vemos que, por exemplo, nos semestres 2012.2, 2016.1 e 2016.2 o percentual foi de 50%, entretanto os respectivos valores absolutos foram 3, 5 e 10. Como cada turma tem um contexto próprio e quantidade diferentes de alunos, não há um comportamento regular, tendo seu máximo (12 alunos ou 48% da turma) em 2014.2, o que mostra que a metodologia não conseguiu diminuir a desistência. Com o intuito de demonstrar a significância da evasão para as reprovações, nessa mesma figura, também incluímos a curva com a quantidade absoluta de reprovações por semestre. Na maioria dos casos o quantitativo de reprovações é maior que desistências, tendo igualdade apenas em 2012.2 e 2016.1, contudo destacamos o grande impacto da evasão para reprovações.

Nesse estudo também foi observado os dados sobre as aprovações durante os semestres (Fig. 7). Nos gráficos em barra são comparados o percentual de aprovação real (desconsiderando os desistentes) e o percentual aprovação absoluta (com desistentes), e os respectivos valores quantitativos (absolutos) em cada semestre (na curva em cinza). Em todos os semestres, o percentual de aprovação real é sempre maior que com os desistentes. Percebe-se também que nos semestres depois da metodologia possui uma taxa de aprovação quase sempre acima de 50%. Adicionalmente incluímos uma curva com o quantitativo de alunos em recuperação em cada semestre (curva em verde), a qual tem um comportamento bem semelhante a curva de aprovação, o que sugere que a grande maioria que vai até o fim do curso (mesmo em recuperação) consegue aprovação.

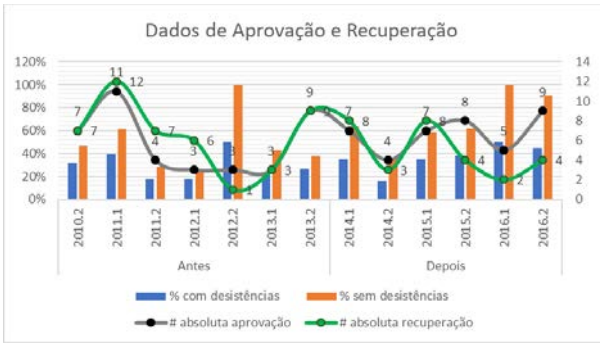

Fig. 7. Dados sobre aprovação e recuperação em OAC

O mesmo estudo foi realizado com as turmas de SO. Entretanto a aplicação da metodologia começou a ser aplicada apenas nos últimos quatro semestres, logo, de 2010.2 a 2014.2 corresponde a antes da metodologia e de 2015.1 a 2016.2 aos semestres de aplicação da metodologia. Na Fig. 8 é apresentado o gráfico em barras correspondentes ao percentual de desistência com relação ao total de alunos, a curva em cinza apresenta a quantidade absoluta de alunos desistentes e a curva em azul o quantitativo de alunos reprovados.

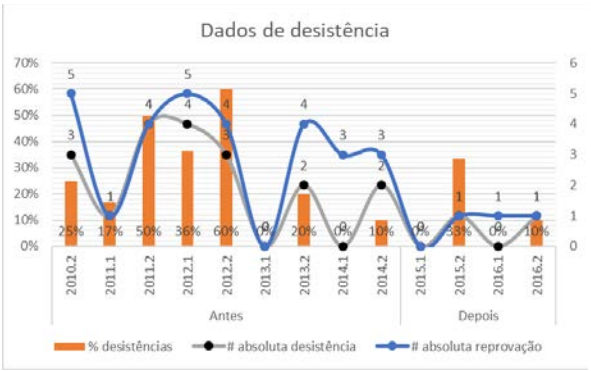

Fig. 8. Dados sobre desistência em SO antes e depois da metodologia

As turmas de SO são relativamente menores que as de OAC, logo, desistências de 4 alunos (em 2011.2) e 3 (em 2012.2) correspondem a 50% e 60% respectivamente. Observando a curva de reprovações (em azul), há muito mais pontos de interseção com a curva das desistências (em cinza) em comparação as mesmas curvas de OAC. Em SO, essas curvas são muito próximas (em 6 semestres elas são iguais), o que sugere também que há um grande impacto da evasão sobre as reprovações. Também podemos destacar que os semestres depois da metodologia estão entre os de menor desistência e reprovações.

Em relação aos dados de aprovação e recuperação, a Fig. 9 apresenta os gráficos em barras com o percentual de aprovação incluindo as desistências no total de alunos e sem os

desistentes. Como os índices de desistências são menores em SO que OAC, os percentuais (com e sem desistentes) da Fig. 9 são bem próximos. A quantidade absoluta de aprovações é apresentada na curva em cinza e de alunos em recuperação na curva em verde. E mais uma vez se observa a semelhança das duas curvas com pontos muito próximos ou iguais, mesmo em semestres como 2014.2 com 11 alunos em recuperação (em torno de 80%) ainda assim houveram 17 aprovados (quase 100%). Nos semestres depois da metodologia temos um alto percentual de aprovação e com baixos índices de recuperação, mas pela quantidade pequena de amostragem não é possível realizar uma conclusão precisa da efetividade da metodologia.

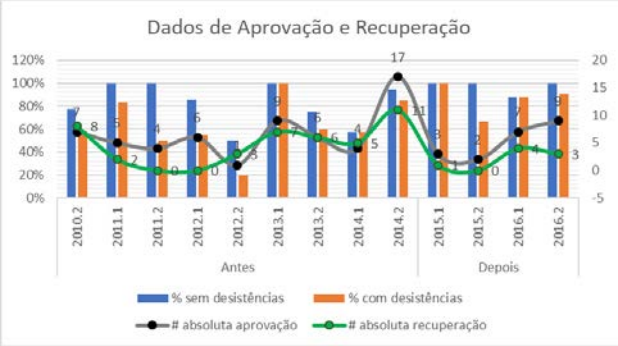

Fig. 9. Dados sobre aprovação e recuperação em SO

Com relação ao impacto da metodologia sobre as notas, consideramos apenas os alunos que cursaram as disciplinas do início ao fim (excluindo os desistentes). A TABELA 2 resume o estudo das notas antes e depois da metodologia em OAC, com a média, desvio padrão, notas máximas e mínimas e o tamanho amostral. Logo após essa tabela, apresentamos na Fig. 10, a distribuição normal e frequência dessas notas.

TABELA 2 - ESTUDO DAS NOTAS ANTES E DEPOIS DA METODOLOGIA EM OAC

	ANTES	DEPOIS
Média	3,6	4,2
Desvio Padrão	2,594804574	2,808834
Nota Mínima	0,2	0,3
Nota Máxima	10,0	9,8
Tamanho amostral	112	75

No gráfico em barras da Fig. 10, percebemos que a maior frequência de notas (antes e depois) está entre 0 e 1, mas com um quantitativo consideravelmente menor em "Depois". A menor frequência nos dois casos é de nota 10, mas com aumento em "Depois". Após a metodologia houve diminuição na frequência da menores notas e aumento das maiores. Nessa mesma figura é apresentada a distribuição normal dessas frequências calculada por uma função densidade massa (FDM) com 100 pontos tanto para "Antes" como para "Depois". Uma característica dessa distribuição é seu pico na média, então a curva "FDM Depois" é ligeiramente deslocada para direita e com decaimento suave em relação as maiores notas. Enquanto que a "FDM Antes" tem seus maiores valores em relação as menores notas e um decaimento mais abrupto nas maiores notas.

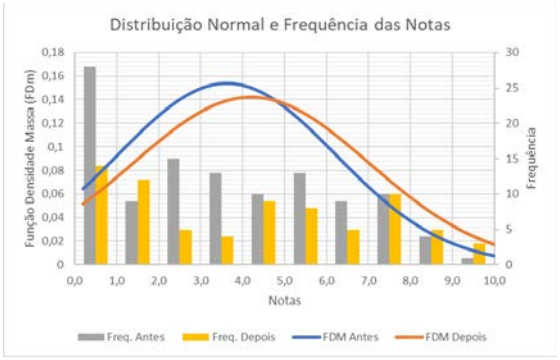

Fig. 10. Distribuição normal e frequência das notas em OAC

É interessante observar que o maior desvio padrão para "Depois" representa esse maior espalhamento das frequências pelas diversas notas, o que é positivo uma vez que a concentração maior era nas menores notas em "Antes".

O mesmo estudo realizado nas turmas de SO são apresentados na TABELA 3 e na Fig. 11.

TABELA 3 - ESTUDO DAS NOTAS ANTES E DEPOIS DA METODOLOGIA EM SO

	ANTES	DEPOIS
Média	6,1	7,4
Desvio Padrão	2,222790972	1,317537385
Nota Mínima	0,7	4,5
Nota Máxima	10,0	9,4
Tamanho amostral	74	21

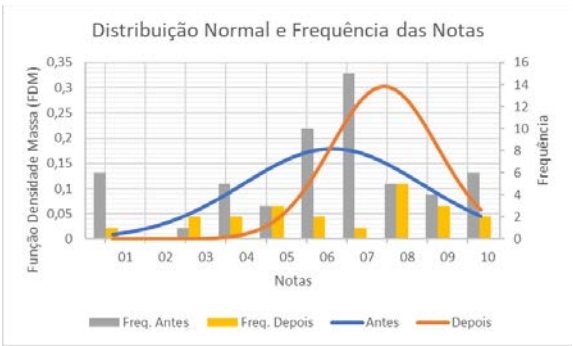

Fig. 11. Distribuição normal e frequência das notas em SO

Antes da metodologia temos a menor e a maior notas, mas uma menor média e um maior desvio padrão. Mesmo "Antes" ter um tamanho amostral maior, ainda assim podemos inferir uma melhoria "Depois" uma vez que essas curvas também foram relativizadas com distribuição normal por meio da FDM com 100 pontos. As maiores frequências de "Antes" estão nas notas entre 5 e 7 e confirmado na curva de distribuição normal. Quando analisamos as notas de "Depois", as menores frequências são nas menores notas e uma curva de distribuição mais concentrada entre as notas 7 e 9. Em SO vemos mais claramente um impacto positivo da metodologia sobre as notas dos alunos.

Por fim, a metodologia foi avaliada pelos alunos que cursaram as duas disciplinas. Cada aluno só pode responder o questionário uma vez, mesmo que já tenham cursado a disciplina várias vezes. O questionário é formado por 12 questões de múltipla escolha, tendo cada questão 5 opções de respostas, exceto a última com 2 opções. As perguntas do questionário são listadas na Tabela 4. Sendo as questões 2, 3, 4 e 11 com respostas individuais para cada disciplina. As questões 6 e 7 são exclusivas da disciplina OAC e as 8, 9 e 10 exclusivas de SO.

TABELA 4. QUESTIONÁRIO DA AVAIAÇÃO PELOS ALUNOS

Q1	Qual sua opinião sobre a interação entre disciplinas do curso de Ciência da Computação de modo geral?
Q2	Como você vê o uso de atividades práticas (implementações e uso de simuladores) para cada disciplina
Q3	Qual sua opinião geral sobre as atividades realizadas com o MARS para cada disciplina
Q4	Para você, com relação ao aprendizado, as atividades práticas envolvendo o MARS, tiveram que impacto, para cada disciplina
Q5	Qual sua opinião sobre a utilização do MARS (como assembler/simulador do MIPS e como uma ferramenta extensível) para interação das disciplinas de Organização e Arquitetura de Computadores e Sistemas Operacionais?
Q6	Qual sua opinião sobre a atividade de implementação de código assembly no MARS na disciplina de Arquitetura de Computadores?
Q7	Qual sua opinião sobre a atividade de implementação VHDL do MIPS e usando o MARS para gerar o programa do processador na disciplina de Arquitetura de Computadores?
Q8	Qual sua opinião sobre a atividade de implementação de syscall do MARS para criação de processos e troca de processos na disciplina de Sistemas Operacionais
Q9	Qual sua opinião sobre a atividade de implementação de uma tool do MARS para gerenciamento de memória virtual na disciplina de Sistemas Operacionais
Q10	Qual sua opinião sobre a atividade de implementação de uma tool do MARS para tratamento de interrupção externa relativa a um timer na disciplina de Sistemas Operacionais
Q11	O que você acha que mais poderia melhorar nas atividades práticas envolvendo o MARS para cada disciplina
Q12	Para você, com as funcionalidades e extensibilidade do MARS, em quais disciplinas também poderiam haver interação com uso dessa ferramenta?

As opções de respostas foram dispostas de “discordar completamente” até “concordar completamente” com a pergunta (ou de “pior” até “melhor”). As questões 11 e 12 possuem suas particularidades que serão abordadas separadamente. Dessa forma, os pesos das respostas variam de 1 a 5, e a resposta com peso 3 foi usada como “não sei ou não se aplica”. Com base nesses pesos, foi calculada a média das respostas para cada questão (de Q1 a Q10), as quais são sumarizadas na Fig. 12.

Fig. 12. Média das respostas do questionário

A média das respostas, em sua grande maioria está acima de 4, o que indica a aprovação da metodologia por parte dos alunos. As respostas que estão abaixo de 4 foram nas questões 7 e 10, mas ainda assim com nota média acima de 3 (nota dos indecisos). Essas questões indicam respectivamente que o uso do VHDL está entre “adequado” ou “não sei ou não se aplica”; as atividades envolvendo sistemas operacionais em sua grande parte foi respondida como “não sei ou não se aplica”, pois, a metodologia foi aplicada mais recentemente em SO, como na questão 10, e nem todos os alunos a experimentaram.

Os resultados da questão 11 são apresentados na Fig. 13. Nas atividades de OAC, excluindo a maioria que “não sabe ou não se aplica” foi indicado como melhoria a clareza entre teoria e prática, assim como a quantidade desse tipo de atividade. Em SO, a maioria reclamou dos enunciados das atividades, seguida pelo aumento da quantidade também.

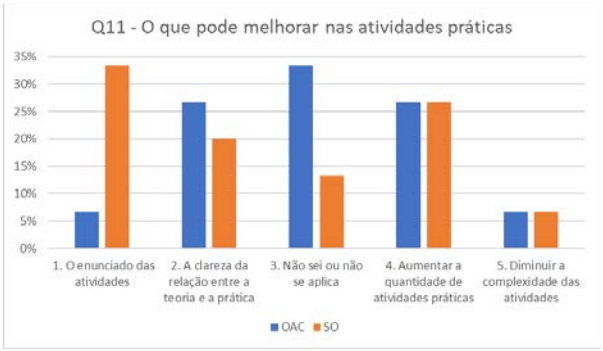

Fig. 13. Respostas da questão 11

A questão 12 avaliou a percepção dos alunos quanto as potencialidades do MARS como ferramenta interdisciplinar. Essa questão permitia-os indicarem em que outras disciplinas essa ferramenta poderia ser usada além das duas em questão. E para surpresa, 100% das respostas acredita que o MARS pode ser usado apenas em OAC e SO.

V. CONCLUSÕES

Neste artigo foi realizado um relato de experiência interdisciplinar, que consiste na aplicação de atividades práticas nas disciplinas de Organização e Arquitetura de Computadores (OAC) e Sistemas Operacionais (SO) em turmas de ciência da computação na UFERSA, durante 13 semestres. A relação entre as atividades nas duas disciplinas é o processador MIPS. Na disciplina de OAC foram realizadas atividades com o simulador MARS, com uma implementação VHDL do MIPS e a ferramenta MIPSFPGA. Na disciplina de SO, as atividades envolvem a extensão das funcionalidades do MARS, desenvolvendo novas *syscalls* para gerenciamento de processo e *tools* para gerenciamento de memória virtual e para gerenciamento de E/S e tratamento de interrupção.

A avaliação dessa metodologia foi realizada por meio da análise estatística dos dados de desistência, de reprovação, de aprovação e de recuperação dos alunos nas turmas das duas disciplinas, na análise do impacto da metodologia nas notas das turmas e por um questionário avaliativo respondido pelos alunos. Pela análise, conclui-se que a taxa de desistência não foi afetada pela metodologia, contudo a taxa de aprovação foi

afetada positivamente assim como o impacto nas notas. A avaliação realizada pelos alunos aprovou a metodologia na maior parte, chamando a atenção para melhorias no desenvolvimento dos enunciados das atividades com VHDL e nas de SO.

Com o uso do processador MIPS e ferramentas relacionadas foi possível mostrar uma metodologia interdisciplinar na qual pode se estender para outras disciplinas, além de OAC e SO, contrariando a expectativa dos alunos. Entre outras disciplinas, essas ferramentas também poderiam ser usadas em compiladores, microprocessadores & microcontroladores ou sistemas embarcados e software básico. Em compiladores, seria possível desenvolver os analisadores léxico, sintático e semântico de uma linguagem de alto nível para traduzi-la para o *assembly* do MIPS, integrando-o ao editor do MARS. Em microprocessadores & microcontroladores ou sistemas embarcados é possível desenvolver ferramentas (*tools*) para o MARS que simulem dispositivos externos como sensores e atuadores os quais se comunicam com o processador MIPS. Para software básico, além do montador que já existe no MARS poderia ser incluído um *linker* e *loader* os quais seriam acionados no momento da simulação, permitindo simulações de ligação dinâmica ou soluções nos problemas de relocação.

Ainda há trabalhos futuros que devem ser implementados nas próximas turmas de OAC e SO. Em OAC é possível utilizar o MARS no estudo de dispositivos de E/S, por meio de programação *assembly* para controlar as *tools* já existentes com esse propósito (*Bitmap Display* e *Keyboard and Display MMO Simulator*). Também em OAC pode ser usada a *tool* BHT para ilustrar o previsor de salto. Em SO, planeja-se o desenvolvimento de *syscalls* para as funções de semáforo; implementação de diversos algoritmos de escalonamento e comparação de eficiência entre eles; atividades relacionadas ao sistema de arquivo.

AGRADECIMENTOS

Agradeço a todos os alunos que responderam gentilmente o questionário, mesmo aqueles que já cursaram a disciplina há bastante tempo.

REFERENCES

[1] W. L. de Oliveira, G. H. Arruda, and R. A. Bittencourt, “Uso do método PBL no ensino de arquitetura de computadores,” 2007.

[2] Wagner L. A. de Oliveira, Anfranserai M. Dias, Antonio L. Apolinário Júnior, Angelo A.Duarte, and Tiago de Oliveira, “Ensino de Arquitetura de Computadores: Uma Abordagem Utilizando a Metodologia de Aprendizagem Baseada em Problemas,” in *Arquitetura de Computadores: educação, ensino e aprendizado*, Carlos Augusto Paiva da Silva Martins, Philippe Olivier Alexandre Navaux, Rodolfo Jardim de Azevedo, and Sérgio Takeo Kofuji, Eds. SBC, 2012.

[3] R. de Oliveira Duarte and P. F. Donoso-Garcia, “Ensino Prático de Projeto de Processadores Segundo uma Metodologia de Ensino-Aprendizagem baseada em Projetos na Escola de Engenharia da UFMG.”

[4] Z. Gao, G. Xue, G. Dai, and X. Wei, “Applying Two New Methods to the Teaching of Computer Architecture,” in *2010 10th IEEE International Conference on Computer and Information Technology*, 2010, pp. 2109–2113.

[5] P. H. Penna and H. C. Freitas, “Análise e Avaliação de Simuladores de Sistemas Completos para o Ensino de Arquitetura de Computadores.”

[6] K. Vollmar and P. Sanderson, “A MIPS Assembly Language Simulator Designed for Education,” *J. Comput. Sci. Coll.*, vol. 21, no. 1, pp. 95–101, 2005.

[7] K. Vollmar and P. Sanderson, “MARS: An Education-Oriented MIPS Assembly Language Simulator,” in *ACM SIGCSE*, 2006, pp. 239–243.

[8] A. L. Torres and A. V. Brito, “Extensão do Ptolemy para o ensino de Organização e Arquitetura de Computadores,” in *Workshop sobre Educação em Arquitetura de Computadores*, 2011, p. 13.

[9] J. Ge, B. Ye, X. Fei, and B. Luo, “A Novel Practical Framework for Operating Systems Teaching,” in *2009 International Conference on Scalable Computing and Communications; Eighth International Conference on Embedded Computing*, 2009, pp. 596–601.

[10] J. Hill, C. K. Ray, J. R. Blair, and C. A. Carver Jr, “Puzzles and games: addressing different learning styles in teaching operating systems concepts,” in *ACM SIGCSE Bulletin*, 2003, vol. 35, pp. 182–186.

[11] J. Nieh and C. Vaill, “Experiences teaching operating systems using virtual platforms and linux,” *SIGCSE Bull*, vol. 37, no. 1, pp. 520–524, 2005.

[12] H. Yi-Ran, Z. Cheng, Y. Feng, and Y. Meng-Xiao, “Research on teaching operating systems course using problem-based learning,” in *2010 5th International Conference on Computer Science & Education*, 2010, pp. 691–694.

[13] R. A. Hexsel and R. Carmo, “cMIPS–uma Ferramenta Pedagógica para o Estudo de Arquitetura,” in *WEAC’13: Workshop sobre Educaçao em Arquitetura de Computadores*, 2013, pp. 1–4.

[14] A. M. Oliveira, J. R. B. Garay, A. C. L. Rodrigues, J. F. Justo, and S. T. Kofuji, “A Teaching Methodology Based On The ALU 8bit RISC Design VLSI Full Custom for Classes on Computer Architecture and Digital Electronic,” *Int. J. Comput. Archit. Educ. IJCAE*, vol. 2, no. 1, pp. 25–28, 2013.

[15] C. A. Zeferino, A. L. A. Raabe, P. V. Vieira, and M. C. Pereira, “Um Enfoque Interdisciplinar no Ensino de Arquitetura de Computadores,” *C Martins P Navaux R Azevedo Kofuji Arquitetura Comput. Educ. Ensino E Aprendizado*, 2012.

[16] P. V. Vieira, A. L. A. Raabe, and C. A. Zeferino, “Bipide: Ambiente de desenvolvimento integrado para utilização dos processadores bip no ensino de programação,” *XX SBIE*, 2009.

[17] L. S. G. de Carvalho and F. G. Nakamura, “Práticas de Ensino na Disciplina de Circuitos Lógicos.”

[18] “KTechlab.” [Online]. Available: http://sourceforge.net/projects/ktechlab/.

[19] Cisco Systems Inc., “Cisco Binary Game.” [Online]. Available: http://forums.cisco.com/CertCom/game/binary_game_page.htm.

[20] D. Patterson and J. Hennessy, *Organização e projeto de computadores – a interface hardware software.*, 4th ed. Campus, 2013.

[21] A. S. Tanenbaum, “MINIX 3.” [Online]. Available: http://www.minix3.org/.

[22] A. S. Tanenbaum, *Sistemas Operacionais. Projeto e Implementação*, 3rd ed. Bookman, 2008.

[23] D. Harris and S. Harris, *Digital Design and Computer Architecture*, 2nd ed. 2012.

[24] P. Sanderson and K. Vollmar, “An Assembly Language I.D.E. To Engage Students Of All Levels,” presented at the 2007 CCSC, 2007.

[25] D. Patterson and J. Hennessy, *Computer Organization and Design: the Hardware/Software Interface*, 3rd ed. San Francisco: Elsevier, 2005.

[26] Altera, “DE2-115 Development and Education Board,” 2013. [Online]. Available: https://www.altera.com/solutions/partners/partner-profile/terasic-inc-/board/altera-de2-115-development-and-education-board.html.

[27] B. Wanderley Neto, *Arquitetura de Computadores: A visão do software*. CEFET-RN, 2005.

[28] M. R. D. Araujo, F. L. C. Padua, F. V. Andrade, and F. L. Correa-Junior, “MIPS X-Ray: A MARS Simulator Plug-in for Teaching Computer Architecture.,” *iJES*, vol. 2, no. 2, pp. 36–42, 2014.

[29] J. C. Penha, G. Fontes, and R. Ferreira, “MIPSFPGA - Um Simulador MIPS Incremental com Validação em FPGA,” *Int. J. Comput. Archit. Educ. IJCAE*, vol. 5, no. 1, pp. 19–25, 2016.

.