Capítulo 2

DrMIPS -Ferramenta de Apoio ao Ensino e Aprendizagem de Arquitectura de Computadores

Bruno Nova
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal

António Araújo e João C. Ferreira
INESC TEC e Faculdade de Engenharia da Universidade do Porto
Porto, Portugal

***Title* – DrMIPS - Tool to Support Computer Architecture Teaching and Learning**

***Abstract* –Computer architecture is an important subject in informatics and electrical engineering courses. However, students display some difficulties in this subject, mainly due to the lack of educational tools that are intuitive, versatile and with a graphical interface. Existing tools are not adequate enough or are very specific. In this paper, an educational MIPS simulator, DrMIPS, is described. This tool simulates the execution of an assembly program on the CPU and displays the datapath graphically. Registers, data memory and assembled code are also displayed and a performance analysis mode is also included. Both unicycle and pipeline implementations are supported and the CPUs and their instruction sets are configurable. The tool is open source and is currently available for PCs and Android devices, and is fairly intuitive and versatile on both platforms.**

***Keyworks* –MIPS; Simulator; Computer Architecture teaching**

***Resumo* – Arquitectura de Computadores é uma disciplina importante nos cursos de engenharia informática e electrotécnica. Contudo, os estudantes demonstram algumas dificuldades nesta disciplina, principalmente devido à falta de ferramentas educativas de apoio que sejam intuitivas, versáteis e com interface gráfica. As ferramentas existentes não são suficientemente adequadas ou são demasiado específicas. Neste artigo é descrito um simulador educativo do MIPS, o DrMIPS. Esta ferramenta simula a execução de um programa em *assembly* no CPU e mostra o caminho de dados graficamente. Registos, memória de dados e código máquina são também mostrados e um modo de análise de desempenho é também incluído. Ambas as implementações, uniciclo e *pipelined*, são suportadas e as versões do CPU e respectivas instruções são configuráveis. A ferramenta tem o código fonte aberto e está actualmente disponível para PCs e dispositivos Android, e é bastante intuitiva e versátil em ambas as plataformas.**

***Palavras-chave* –MIPS; Simulador; Ensino de Arquitectura de Computadores**

I. INTRODUÇÃO

A. Motivação

A Arquitectura de Computadores é uma disciplina importante no plano de estudos dos cursos de Engenharia Informática e Engenharia Electrotécnica, como o Mestrado Integrado em

Inmaculada Plaza y Martín Llamas Nistal, editores
TICAI 2013-2014: TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-690-9

Engenharia Informática e Computaçãoe o Mestrado Integrado em Engenharia Electrotécnica e de Computadores daFaculdade de Engenharia da Universidade do Porto (FEUP). Nesta disciplina os estudantes adquirem os fundamentos sobre computadores e seus processadores, aprendendo tópicos como representação de dados, circuitos digitais, composição de um CPU, programação em *assembly* e desempenho de processadores.

Muitos estudantes demonstram dificuldades em entender vários assuntos desta disciplina, como processadores com *pipeline* e cálculo do desempenho de um processador. Uma razão para estas dificuldades está na ausência de ferramentas que possam,num ambiente integrado,apoiaro tratamento de vários tópicos de processadores de uma forma orientada para o ensino. Especificamente, detetou-se a falta de ferramentas que permitam aos estudantes ver graficamente a composição do caminho de dados de um CPU, consultar informações detalhadas sobre os dados à entrada e à saída de cada bloco funcional e observar os estados dos sinais de controlo para cada instrução executada pelo CPU, entre outras informações.

Existem várias ferramentas criadas para simular a operação de um CPU, e algumas delas contêm interfaces gráficas para mostrar o caminho de dados do CPU. Porém, a maioria delas é pouco adequada para fins educativos, são difíceis de usar e entender ou são direccionadas para um problema específico,sendo pouco versáteis.

B. Objectivos

O objectivo principal do trabalho apresentado foi criar uma ferramenta de apoio ao ensino e aprendizagem de arquitectura de computadores. Esta ferramenta educativa é um simulador do MIPS [1], que é um processador bastante conhecido na comunidade académica de arquitectura de computadores e também um dos processadores mais usados no ensino de disciplinas de arquitectura de computadores em universidades [2].

Este simulador foi desenvolvido no âmbito de uma Dissertação de Mestrado do MIEIC na FEUP.O seu desenvolvimento foi baseado nos seguintes requisitos:

- Permitir a configuração e parametrização do CPU;
- Permitir a simulação passo-a-passo de programas MIPS em *assembly*;
- Mostrar graficamente o caminho de dados e os valores nas entradas e saídas de cada componente;
- Simular ambas as versões uniciclo e *pipelined* do CPU;
- Ter um "modo de desempenho" que mostre as latências e o caminho crítico do CPU;
- Ser versátil, intuitivo e simples de utilizar.

Em termos de apoio, pretende ajudar os estudantes a melhor entender:

- A composição e funcionamento de um caminho de dados "simples";
- Como as instruções são codificadas;
- Os valores dos sinais no caminho de dados;
- Blocos e sinais relevantes para cada instrução;
- *Pipelining*, conflitos, atalhos e protelamentos,
- Medição de desempenho e identificação do caminho crítico.

A ferramenta foi desenvolvida principalmente para computadores pessoais, mas uma versão para dispositivos Android, especialmente *tablets*, também foi criada.

O resto deste artigo está organizado em secções que a seguir se descrevem. A Secção II apresenta o estado da arte descrevendo os simuladores educativos mais relevantes. A Secção III explica como o código da ferramenta foi estruturado e como a lógica de simulação foi implementada. A Secção IV apresenta e detalha a implementação da interface gráfica. Por último, a Secção V refere as conclusões e discute possível trabalho futuro.

II. ESTADO DA ARTE

Tal como referido anteriormente, existem várias ferramentas que simulam o funcionamento de um CPU, e algumas até apresentam a composição do caminho de dados graficamente. Contudo, a maioria delas não são vocacionadas para instruir estudantes em disciplinas de arquitectura de computadores, são demasiado difíceis para um estudante usar ou focam-se num problema muito específico à margem do pretendido. Esta secção apresenta os simuladores

Inmaculada Plaza y Martín Llamas Nistal, editores
TICAI 2013-2014: TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-690-9

educativos mais relevantes, bem como as suas vantagens e desvantagens.

O QtSPIM (anteriormente chamado SPIM) [3] é um simulador com o código-fonte aberto, escrito em C++ e Qt, que executa programas MIPS32. Foi bastante utilizado, tanto no ensino como na indústria [4], e suporta um grande número de instruções MIPS, incluindo chamadas de sistema e operações de vírgula flutuante [5]. A ferramenta é boa para depurar programas MIPS em *assembly* e é razoavelmente intuitiva, mas simula somente a versão uniciclo do CPU. Também não permite a visualização gráfica do caminho de dados nem possui um editor de código integrado.

O simulador MARS [4], desenvolvido em Java, é usado em disciplinas de arquitectura de computadores em muitas universidades por todo o mundo. Ele simula a execução de um programa MIPS em *assembly*, mostrando os resultados e os valores dos registos e memória no ecrã. A simulação pode ser executada passo-a-passo ou de uma só vez. O conjunto de instruções suportado inclui operações de vírgula flutuante, chamadas de sistema e várias pseudo-instruções. O MARS é também um IDE que inclui um editor com realce de sintaxe e muitos tópicos de ajuda. A ferramenta é bastante boa para simular e depurar programas MIPS em *assembly*. Contudo, apenas simula a versão uniciclo do CPU e não mostra o caminho de dados graficamente. O MARS suporta *plugins*. Um destes é o MIPS X-Ray [6], que apresenta graficamente o caminho de dados uniciclo do MIPS, assim como as ligações e componentes relevantes para as instruções.Aexecução destas é apresentada recorrendo a animações.

O ProcSim [7] é uma ferramenta desenvolvida em Java que simula o CPU uniciclo MIPS R2000. O código *assembly* é executado e mostrado graficamente como uma animação no caminho de dados. A ferramenta inclui vários caminhos de dados diferentes e o utilizador pode criar outros. Um editor de código muito simples é também fornecido. O ProcSim fornece uma boa visualização do caminho de dados. Contudo, suporta um pequeno conjunto de instruções MIPS e somente um componente pode enviar mensagens ao mesmo tempo durante a simulação, apresentando as animações sequencialmente por componente, enquanto num processador real os componentes funcionam concorrentemente [8]. Para além disso, não suporta caminhos de dados *pipelined*.

O MIPS-Datapath [9], desenvolvido em C++, é um programa de código-fonte aberto que simula um conjunto de instruções MIPS e apresenta a execução graficamente no caminho de dados. Ele consegue simular não só um caminho de dados uniciclo mas também a versão*pipelined*, com ou sem atalhos. As instruções são executadas passo-a-passo e as ligações relevantes para a instrução seleccionada são realçadas. Um editor de código muito simples é também fornecido. O MIPS-Datapath permite visualizar como cada instrução é executada pelo processador. Porém, ele suporta um conjunto de instruções muito limitado, não suporta protelamentos no *pipeline* e não permite que o caminho de dados seja configurado.

O WebMIPS [10] é um simulador educativo do MIPS que pode ser executado através de um navegador Web e, portanto, a partir de qualquer sistema, sem instalação. Foi escrito em ASP e simula um *pipeline* de cinco etapas com resolução de conflitos, tendo sido usado numa disciplina introdutória de arquitectura de computadores em Itália. A aplicação fornece um editor simples de código,podendo este ser executado passo-a-passo, mostrando também uma representação gráfica do caminho de dados. Apesar de ser um bom simulador educativo, tem algumas lacunas. A simulação da versão uniciclo do MIPS não é possível e a representação gráfica do caminho de dados é estática, mostrando os dados nas entradas/saídas apenas após um clique.

O EduMIPS64 [11] é um simulador educativo que executa programas MIPS64. A ferramenta foi usada em alguns cursos de graduação para a avaliar e os resultados foram positivos, tanto em termos de percentagem de sucesso [12] como em termos de apreciação dos estudantes [13]. Este simulador foi baseado no WinMIPS64 [14] e no WinDLX e foi desenvolvido em Java. A ferramenta simula um CPU MIPS64 com 5 etapas de *pipeline*, suporta um número bastante razoável de instruções, incluindo operações em vírgula flutuante, chamadas de sistema e resolução de conflitos. A interface é intuitiva, mas nenhum editor de código é fornecido, não suporta a simulação do CPU uniciclo e não possui uma representação detalhada do caminho de dados.

Inmaculada Plaza y Martín Llamas Nistal, editores
TICAI 2013-2014: TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-690-9

A Tabela I sumariza as ferramentas apresentadas. *PS*representa o ProcSim, *MD*representa o MIPS-Datapath, *WM*representa oWebMIPS e *EM*representa o EduMIPS64. Adicionalmente, *Parc.* significa que a ferramenta suporta a funcionalidade parcialmente e *Simpl.*significa, para a funcionalidade *Caminho dados gráf.*, que a ferramenta apresenta um diagrama de blocos muito simples. A tabela mostra que seria necessário utilizar mais do que uma ferramenta para cobrir os vários tópicos de arquitectura de computadores, o que seria pouco prático e intuitivo para estudantes e professores.

Para Android foi encontrado apenas um simulador do MIPS, o Assembly Emulator [15], sendo um simulador relativamente básico e que, durante a simulação, apenas apresenta as instruções que estão a ser executadas e os valores dos registos. Como tal, uma versão do simulador desenvolvido para *tablets* Android constitui um importante contributo, tendo em consideração que os *tablets* Android estão-se a tornar bastante populares [16], [17].

TABELA I
COMPARAÇÃO DAS FERRAMENTAS APRESENTADAS

	SPIM	MARS	PS	MD	WM	EM
Código aberto	Sim	Sim	Não	Sim	Sim	Sim
Editor de código	Não	Sim	Sim	Sim	Sim	Não
Realce de sintaxe	Não	Sim	Não	Não	Não	Não
Uniciclo	Sim	Sim	Sim	Sim	Não	Não
Pipeline	Não	Não	Não	Parc.	Sim	Sim
Vírgula flutuante	Sim	Sim	Não	Não	Não	Sim
Chamadas de sistema	Sim	Sim	Não	Não	Não	Sim
Editar dados na execução	Sim	Sim	Não	Não	Não	Sim
Caminho dados gráf.	Não	Não	Sim	Sim	Parc.	Simpl.
Config. cam. dados	Não	Não	Sim	Não	Não	Não
Escrito em	C++,Qt	Java	Java	C++	ASP	Java

III. IMPLEMENTAÇÃO DO SIMULADOR

O simulador DrMIPS fornece várias versões de caminhos de dados do MIPS, baseados em [1], incluindo o caminho de dados uniciclo com algumas variantes simplificadas e o caminho de dados *pipelined*, com ou sem resolução de conflitos. Estes CPUs podem ser criados e configurados especificando num ficheiro as propriedades de todos os componentes e as ligações entre eles. Além dos caminhos de dados, os respectivos conjuntos de instruções podem também ser configurados, e outros podem ser criados, especificando as propriedades dos diferentes tipos de instruções, instruções e pseudo-instruções e o que elas fazem. Operações de vírgula flutuante e chamadas de sistema não são actualmente suportadas.

O simulador foi implementado não só para PC mas também para dispositivos Android, especialmente *tablets*. Por essa razão, o simulador foi desenvolvido em Java. Isto tornou a adaptação de código da versão para PC para a versão Android e vice-versa fácil, visto o Android também usar Java, e torna também a versão para PC compatível com a maioria dos sistemas operativos sem esforço adicional. A interface gráfica suporta também múltiplos idiomas. De momento, apenas Português e Inglês estão disponíveis. Quanto ao ambiente de desenvolvimento, para a versão para PC foi escolhido o Netbeans, uma vez que facilita a criação de interfaces gráficas; já para a versão para Android foi escolhido o Eclipse, visto ser o recomendado pela Google [18].

Para facilitar o desenvolvimento de ambas as versões, o código foi dividido em duas partes: a lógica de simulação e a interface gráfica. Com esta divisão, apenas a interface gráfica depende da plataforma (PC ou Android), enquanto a lógica de simulação é exactamente a mesma em ambas. A Fig. 1 mostra um diagrama de classes UML do simulador simplificado.

O código da lógica de simulação corresponde ao pacote `simulator`, enquanto o código da interface gráfica corresponde ao pacote `gui`. O pacote`simulator` é composto por vários sub-pacotes, nomeadamente:

- `mips`: contém o simulador do MIPS;
- `mips.components`: pacote interno que define todos os tipos de componentes;
- `util`: contém algumas classes utilitárias;
- `exceptions`: contém as classes de excepções.

O resto desta secção expõe os detalhes da implementação da lógica de simulação, enquanto a secção IV apresenta os detalhes das interfaces gráficas de ambas as versões.

Inmaculada Plaza y Martín Llamas Nistal, editores
TICAI 2013-2014: TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-690-9

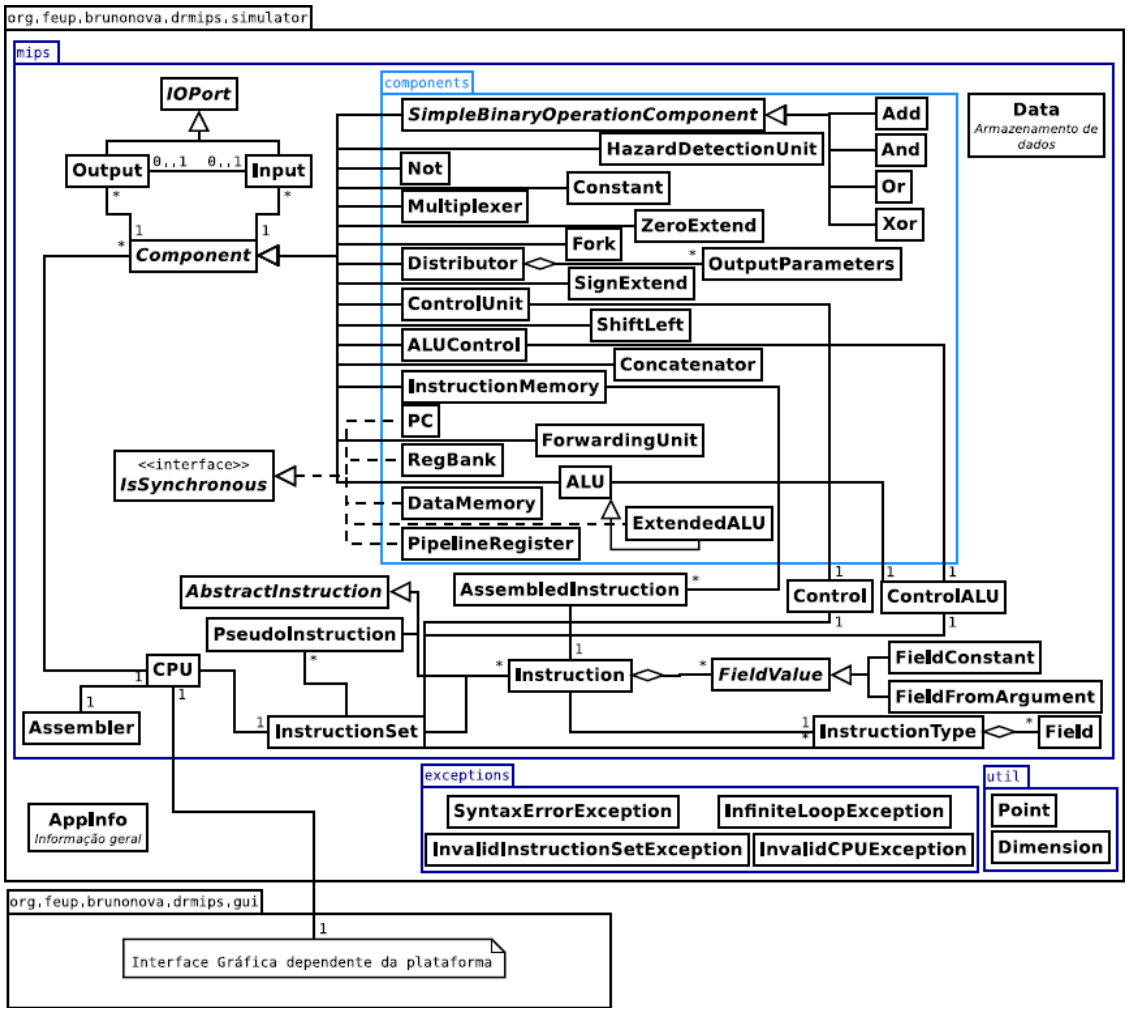


Fig. 1. Diagrama de classes UML do simulador

A. Definição do CPU

Cada CPU é definido num ficheiro JSON. JSON é um formato que pode ser analisado facilmente em Java e em Android. É também fácil de ser lido/escrito peloutilizador e resulta em ficheiros mais pequenos do que outros formatos como XML. Um ficheiro CPU lista os componentes e suas propriedades, as ligações entre eles, o nome de cada registo e o conjunto de instruções usado (descrito na subsecção seguinte)

No código, como mostrado na Fig. 1, a classe `CPU` é a parte central do simulador. Ela permite o acesso a todos os seus componentes, ao conjunto de instruções e ao *assembler*, e é a "interface" que a interface gráfica usa para gerar o código máquina a partir do código fonte, executar instruções, obter valores e latências, etc. É também onde os ficheiros JSON do CPU são carregados e interpretados.

Um CPU é composto por vários componentes. A classe `Component` é a classe base para todos os componentes. Cada componente é derivado desta classe, tendo de definir as suas propriedades e implementar `execute()`, onde o seu comportamento é definido, usando os valores nas entradas para definir os valores correctos nas saídas. Um componente tem entradas e saídas, definidas respectivamente por `Input` e `Output`. Cada saída está ligada a uma entrada de outro componente. Portanto, o CPU é representado por um grafo.

Alguns dos componentes são síncronos, como o contador do programa e o banco de registos, por exemplo. Estes componentes têm um estado interno que apenas pode ser alterado durante a transição do sinal de relógio. Em termos de código, estes componentes implementam a interface `IsSynchronous`, tendo depois de implementar o método `executeSynchronous()`, em que é definido o comportamento síncrono que altera o

seu estado interno. Adicionalmente, para permitir que o utilizador volte atrás durante a execução, os estados internos são guardados em cada ciclo de relógio numa pilha, e alguns métodos adicionais para guardar e restaurar estes estados foram implementados.

Durante a execução de código, cada ciclo de relógio começa pela execução dos comportamentos síncronos dos componentes, prosseguindo depois para a execução dos comportamentos "normais" dos componentes. Durante este processo, os valores nas saídas são actualizados. Quando isto acontece, os novos valores nas saídassão propagados para as entradas ligadas a si, as quais executam o comportamento "normal" dos respectivos componentes, prosseguindo a propagação de dados. Isto acontece apenas quando o novo valor é diferente do anterior, de forma a evitar ciclos infinitos que resultariam desta solução. Isto significa também que o comportamento síncrono dos componentes não pode causar qualquer propagação de dados.

Relativamente ao desempenho do processador, ele também é determinado usando propagação. Cada componente tem a sua latência individual de funcionamento e uma latência acumulada, que é o tempo decorrido desde o início do ciclo até o componente gerar as saídas correctas. As entradas também guardam a latência acumulada. O cálculo das latências acumuladas tem início nos componentes síncronos,sendo propagadas até às entradas que são usadas apenas pelos comportamentos síncronos (como a entrada `WriteData` do banco de registos). O caminho crítico é, posteriormente, calculado para trás a partir da entrada ou entradas com a maior latência acumulada. O simulador também é capaz de determinar o caminho crítico de uma instrução em execução. Neste caso, os sinais de controlo nas entradas de alguns componentes, como multiplexadores e memória de dados, são usados para determinar quais as entradas e os componentes que são irrelevantes para a instrução actual e que por isso não devem entrar no caminho crítico.

Em relação à versão *pipelined* do CPU MIPS, ela tem de ter exactamente cinco etapas e, portando, quatro registos a separá-las. Estes registos, mais o contador do programa, são usados para determinar quais as instruções que se encontram em cada etapa do *pipeline*, verificando também os seus sinais de controlo `Write` e `Flush`. Isto é necessário devido aos conflitos que podem ocorrer. Tanto uma unidade de detecção de conflitos como uma unidade de atalhos foram implementadas, e os seus comportamentos foram baseados em [1].

Em cada ciclo de relógio, cada ligação pode ser marcada como "irrelevante" (a cinza na interface gráfica) ou relevante. Esta decisão é baseada apenas nos valores nas ligações e componentes, e não na instrução. Seria bastante difícil de determinar as ligações e componentes relevantes para cada instrução quando o CPU e até o conjunto de instruções são muito genéricos e configuráveis. Assim, as condições para marcar uma ligação como irrelevante são muito simples: a ligação transporta 1 bit com o valor zero, um protelamento está a ocorrer, a ligação não é seleccionada por um multiplexador, etc.

B. Definição do Conjunto de Instruções

Tal como o CPU, cada conjunto de instruções é definido num ficheiro JSON. Um ficheiro de instruções lista os tipos de instruções, as instruções, as pseudo-instruções e suas propriedades, e também especifica como a unidade de controlo, a ALU e o controlador da ALU funcionam.

A classe principal que define o conjunto de instruções é `InstructionSet` e é acessível através do `CPU`(Fig. 1). Esta classe carrega e interpreta o conjunto de instruções do ficheiro especificado no ficheiro CPU, e fornece acesso a todos os tipos de instruções, instruções, pseudo-instruções e definições de controlo.

Um conjunto de instruções tem várias instruções diferentes. No caso do MIPS, que é uma arquitectura Reduced Instruction Set Computer (RISC), todas as instruções têm o mesmo tamanho. Cada instrução pertence a um tipo e, para o MIPS, isso significa `R`, `I` ou `J`. Os 32 bits que compõe o código da instrução são divididos em campos. Os campos são diferentes para cada tipo, excepto o campo `opcode`, que é comum e formado pelos 6 bits mais significativos. `InstructionType` é a classe que representa essa informação.

O caminho de dados implementado mais completo suporta as seguintes instruções: `nop`,

Inmaculada Plaza y Martín Llamas Nistal, editores
TICAI 2013-2014: TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-690-9

`add`, `sub`, `and`, `or`, `slt`, `addi`, `lw`, `sw`, `beq`, `j`, `nor`, `xor`, `mult`, `div`, `mfhi` e `mflo`. Este caminho de dados inclui também as seguintes pseudo-instruções: `li`, `la`, `move`, `subi`, `sgt`, `neg`, `bge`, `ble`, `b`, `not`, `mul` e `rem`.

As instruções e as pseudo-instruções são representadas pelas classes `Instruction` e `PseudoInstruction`, respectivamente. Cada instrução pertence a um tipo, tem uma mnemónica e define o número e tipo de argumentos, os valores de cada campo (que podem vir de um argumento) e uma descrição simbólica que pode ser vista pelo utilizador. Cada pseudo-instrução tem uma mnemónica e define o número e tipo de argumentos, as instruções na qual é convertida durante a geração do código máquina e uma descrição simbólica. No simulador desenvolvido, cada mnemónica pode pertencer apenas a uma instrução ou a uma pseudo-instrução.

Um conjunto de instruções tem também de definir o que as instruções fazem. Isto é definido pelas classes `Control` e `ControlALU`. A classe `Control` controla o comportamento da unidade de controlo, e contém a informação necessária para produzir os valores dos sinais de controlo para cada `opcode`. A classe `ControlALU` controla o comportamento da ALU e do controlador da ALU.Esta classe contém a informação necessária para o controlador da ALU produzir as saídas para cada combinação de `ALUOp` e campo `func` da instrução, e também a correspondência entre cada possível valor na entrada de controlo da ALU e a respectiva operação.

C. O Assembler

O *assembler*é representado pela classe `Assembler`, e é acessível através do `CPU`, como representado na Fig. 1. Para o código máquina ser gerado a partir do código fonte, a interface gráfica usa o `Assembler` do `CPU` para interpretar o código, consultando o `InstructionSet` do CPU e convertendo as instruções e pseudo-instruções no segmento de texto em instruções máquina, representadas pela classe `AssembledInstruction`, sendo depois carregadas na memória de instruções do CPU. O processo envolve também a interpretaçãodo segmento de dados para inicializar a memória de dados do CPU com os valores pretendidos.

O código introduzido pelo utilizador pode conter erros, devendo ser identificados e mostrados ao utilizador. Quando um erro é encontrado numa instrução, o *assembler* lança uma excepção. Contudo, em vez de parar o processo, a excepção é capturada e adicionada a uma lista de excepções,prosseguindo na instrução seguinte. Com esta técnica, todos os erros presentes no código podem ser mostrados ao utilizador. Todavia, apenas um erro pode ser mostrado por linha.

A implementação do *assembler* é bastante simples, e o processo de geração é feito em dois passos:

1. O *assembler* interpreta o segmento de dados e carrega os valores para a memória de dados. Ao mesmo tempo, as etiquetas são identificadas e as pseudo-instruções são convertidas em instruções no segmento de texto;

2. Todas as instruções são convertidas em instâncias de `AssembledInstruction` equivalentes, sendo depois carregadas para a memória de instruções do CPU.

Em termos de directivas de compilação, quatro directivas são actualmente suportadas:

- `.data`: inicia o segmento de dados, onde a memória de dados é inicializada;
- `.text`: inicia o segmento de texto, onde o código é definido;
- `.word`: declara um ou mais valores a serem armazenados na memória de dados como palavras de 32 bits;
- `.space`: reserva espaço (em bytes) na memória de dados.

IV. IMPLEMENTAÇÃO DA INTERFACE

Esta secção descreve detalhes sobre a implementação das interfaces gráficas para computador e para dispositivos Android. O código da interface gráfica está definido no pacote `gui` (Fig. 1), e é o único pacote que difere entre as versões para PC e Android.

Inmaculada Plaza y Martín Llamas Nistal, editores
TICAI 2013-2014: TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-690-9

A. Versão para PC

A versão para PC é a mais completa. Por omissão, a interface é apresentada com um tema claro e com os seus conteúdos divididos em cinco separadores, como mostrado na Fig. 2. A janela está dividida horizontalmente em duaspartes, permitindo que dois separadores sejam mostrados ao mesmo tempo. Adicionalmente, o utilizador pode mover qualquer separador de um lado para o outro se clicar no título do mesmo e seleccionar a única opção disponível no menu de contexto. Para além disso, o utilizador pode escolher usar um tema escuro, e pode também escolher usar janelas internas em vez de separadores, como mostrado na Fig. 3. Janelas internas são úteis em ecrãs maiores, e as suas posições e tamanhos são recordados ao sair. A interface suporta múltiplos idiomas, Português e Inglês na versão actual, que podem ser escolhidos nos menus.

Os vários separadores ou janelas são:

- Código: contém o editor de código;
- Assemblado:mostra as instruções máquina;
- Caminho de dados:mostra o caminho de dados e as instruções sendo executadas;
- Registos:lista os registos e respectivos valores;
- Memória de dados:mostra os valores na memória de dados.

O DrMIPS fornece um editor de código com realce de sintaxe, conclusão automática de palavras, procurar/substituir, numeração de linhas, desfazer e refazer graças ao componente *RSyntaxTextArea* e à biblioteca *AutoComplete*[19]. As regras de realce de sintaxe e de conclusão de palavras não são "estáticas" e dependem do CPU em uso. Todos os erros presentes no código são indicados ao lado dos números das linhas quando o código fonte é interpretado.

A tabela do código máquina mostra as instruções máquina resultantes e realça a instrução ou instruções a serem executadas pelo CPU. Ao passar o cursor do rato sobre cada instrução o tipo da instrução e os valores dos seus campos são exibidos, como exemplificado na Fig. 3. As tabelas dos registos e memória de dados mostram os respectivos valores, mas também realçam os registos ou endereços acedidos. Os registos e os valores na memória de dados podem ser alterados com um duplo clique sobre os mesmos na respectiva tabela. Todos estes valores podem ser mostrados nos formatos decimal, binário e hexadecimal.

O caminho de dados é a parte mais importante do simulador. Todos os componentes são apresentados como rectângulos e representados internamente por `JPanel`'s. O nome, descrição, valores/latências nas entradas e saídas dos componentes são mostrados numa dica quando o utilizador passa o cursor do rato sobre eles, como mostrado na Fig. 2. As ligações são desenhadas no fundo do caminho de dados e têm cores diferentes quando são irrelevantes ou pertencem ao caminho de controlo. Algumas das entradas e saídas dos componentes apresentam uma pequena dica permanente com o seu valor actual. Essa dica de dados é sempre posicionada abaixo da entrada ou saída e é representada por uma etiqueta do tipo`JLabel`. O caminho de dados é um `JLayeredPane` para permitir que as dicas de dados estejam sempre no topo. O utilizador pode ocultar o caminho de controlo, as dicas de dados e as setas nas ligações, se assim o desejar.

Por opção, a representação do caminho de dados foi a de menortamanho possível, mas garantindo a sua leitura sem dificuldade. Desta forma, uma maior parte do caminho de dados, ou mesmo a totalidade, estará visível. Por esta razão, algumas informações, como os nomes das entradas e saídas, têm de ser omitidas.

O caminho de dados pode também ser apresentado num "modo de desempenho". Neste modo, as ligações que pertencem ao caminho crítico são mostradas a vermelho, as dicas de dados são omitidas e as dicas dos componentes apresentam as suas latências e latências acumuladas. Além disso, um duplo clique no componente, neste modo, permite ao utilizador editar a latência do mesmo. O novo valor não é guardado no ficheiro do CPU, nele permanecendo o valor por omissão.

A instrução ou instruções sendo executadas são apresentadas acima do caminho de dados numa tabela com uma única linha e sem cabeçalho. Usar uma tabela deste modo significa que as colunas provavelmente não estarão alinhadas com os respectivos registos de *pipelined*, mas isso também significa que as colunas estarão sempre visíveis mesmo que o caminho de dados não esteja

Inmaculada Plaza y Martín Llamas Nistal, editores
TICAI 2013-2014: TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-690-9

completamente visível no ecrã, como pode ser visto na Fig. 2.

Finalmente, o utilizador pode consultar algumas estatísticas sobre a simulação. Estas incluem o período e a frequência de relógio, Ciclos Por Instrução (CPI) e número de atalhos e protelamentos ocorridos. A versão para PC inclui também manuais de utilizador e de configuração.

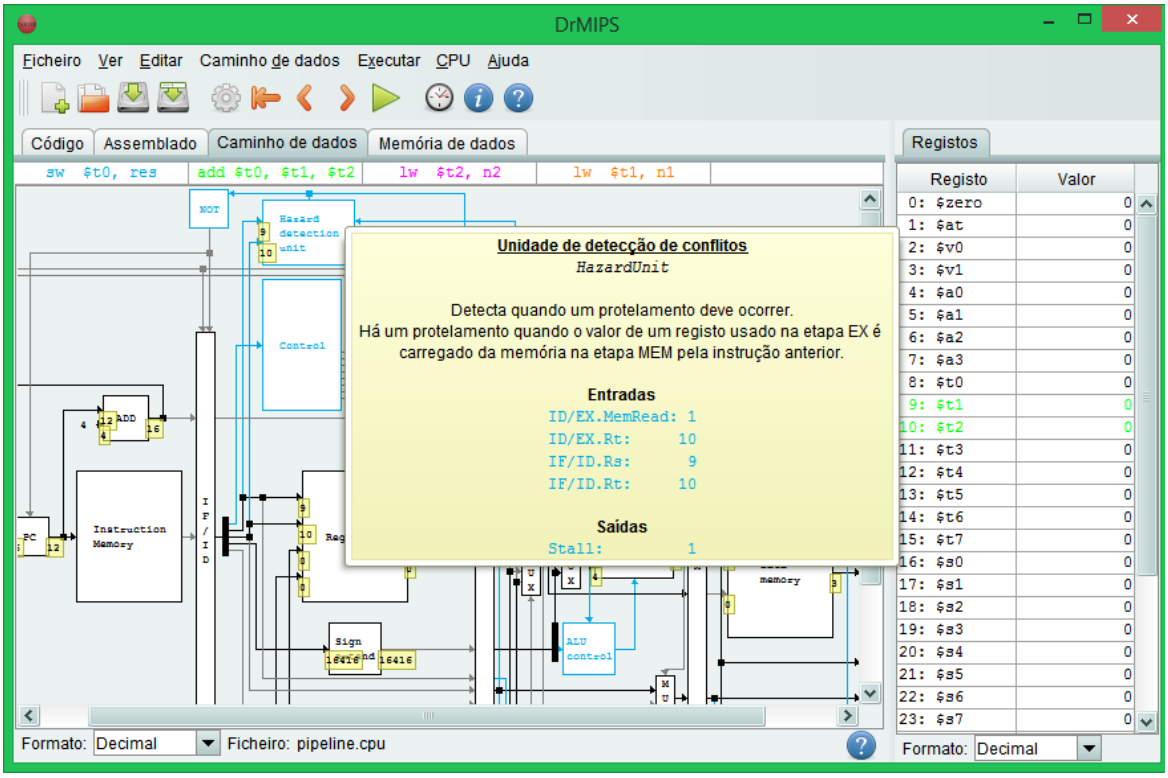


Fig. 2. Interface para PC com as opções por omissão, com a dica de um componente a ser mostrada

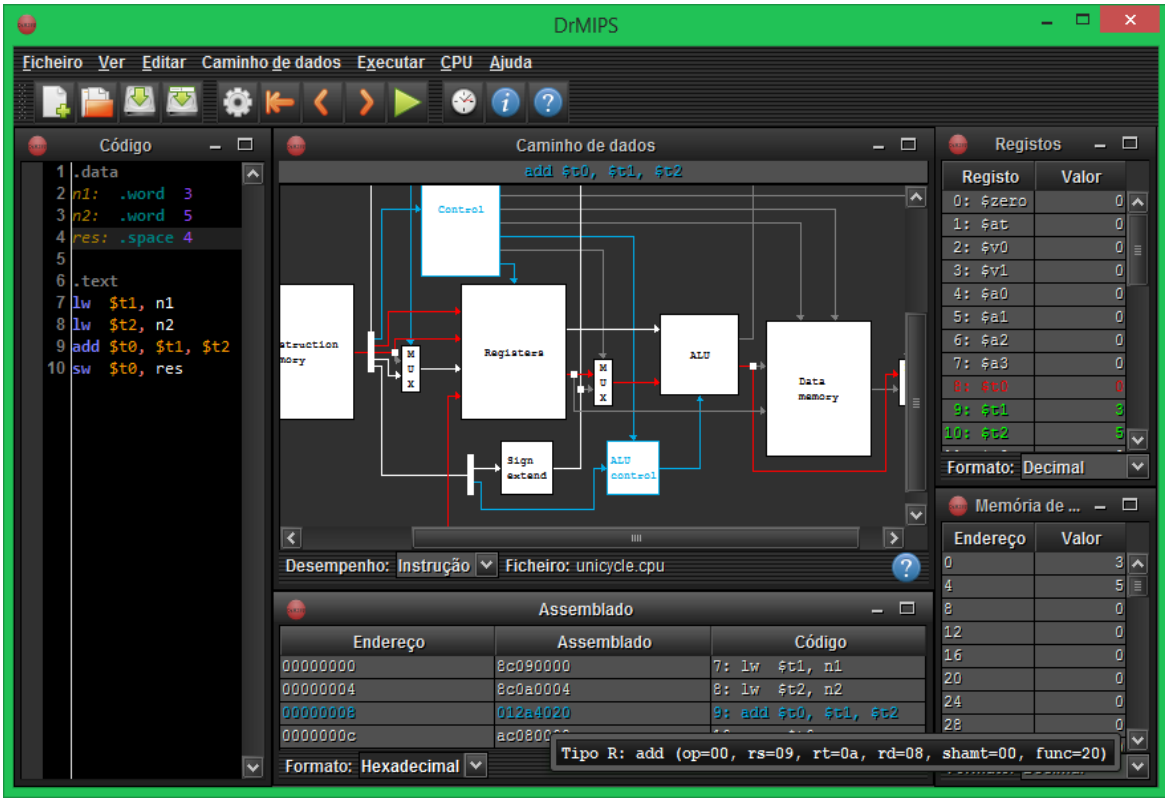


Fig. 3. Interface para PC usando janelas internas e o tema escuro no "modo de desempenho"

B. Versão para Android

A versão para Android é semelhante à versão para PC. Tal como nesta, a aplicação usa por omissão um tema claro, mas um tema escuro está igualmente disponível. A Fig. 4 ilustra a aplicação em execução num *tablet* com Android 4.0.3 usando o tema claro, enquanto a Fig. 5 mostra-a num telemóvel com Android 4.1.2 usando o tema escuro.

A aplicação contém apenas uma actividade e os seus conteúdos são divididos em separadores usando um `TabHost`. Os conteúdos dos vários separadores são definidos em ficheiros de *layout* diferentes, os quais são incluídos no *layout* da actividade sem usar os novos "fragmentos" do Android. A aplicação suporta mudanças de orientação do ecrã sem a perda do seu estado. Actualmente, a aplicação pode ser apresentada em Português ou em Inglês.

O editor de código desta versão é um simples componente do tipo`EditText`. Para além disso, devido ao facto de os programas em*assembly*MIPS utilizarem o cifrão (`$`) para referenciar registos, usar o teclado no ecrã por omissão do Android poderá ser incómodo, visto que o cifrão não se encontra normalmente na sua primeira página. Porém, existem outros teclados disponíveis para Android que podem ser instalados e utilizados, sendo também possível conectar um teclado físico ao dispositivo, o que facilita a escrita de código.

As tabelas com o código máquina, registos e memória de dados são parecidas com as da versão para PC. Contudo, para ver a "dica" de uma instrução no código máquina, o utilizador tem de tocar na instrução em vez de passar o "cursor" sobre a mesma, visto os dispositivos móveis normalmente não disporem de um cursor ou rato. Adicionalmente, para alterar um registo ou um valor na memória de dados, o utilizador tem de tocar e manter pressionado o mesmo na tabela em vez de realizar um duplo clique. Este comportamento é usado noutras partes da interface.

O caminho de dados é mostrado da mesma forma que na versão para PC. É usada uma disposição do tipo`RelativeLayout` para apresentar os componentes nas posições e

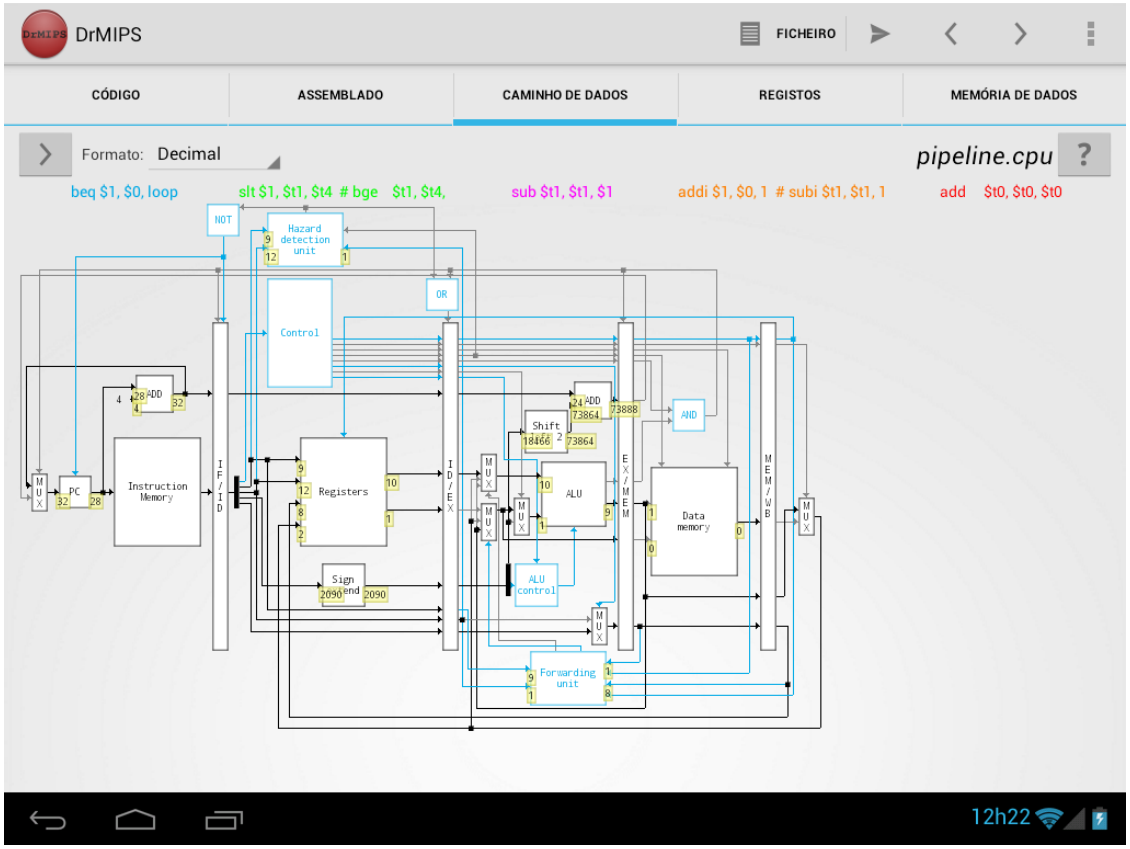


Fig. 4. Interface para Android usando o tema claro num *tablet*

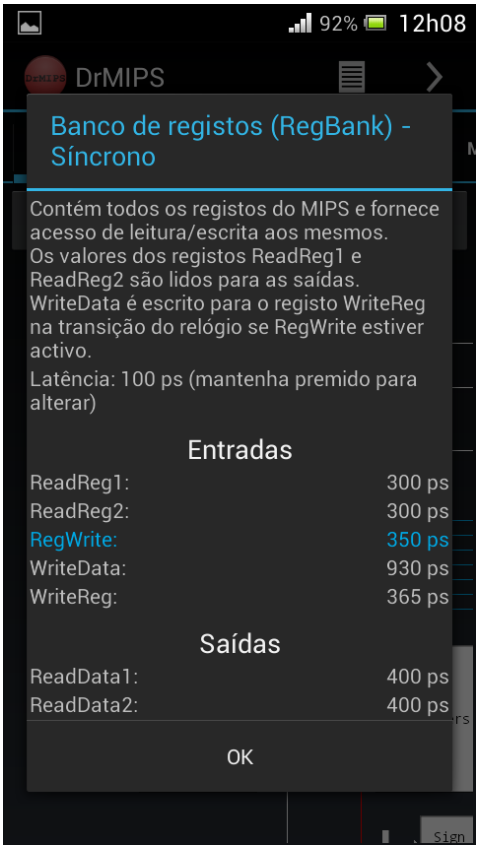


Fig. 5. Interface para Android usando o tema escuro num telemóvel, mostrando os detalhes de um componente no "modo de desempenho"

tamanhos especificados, todos medidos usando *pixels*independentes da densidade, que é uma unidade que permite que os componentes gráficos tenham aproximadamente o mesmo tamanho real em todos os dispositivos. Cada componente é do tipo`TextView` e o seu nome, descrição e valores/latências nas entradas e saídas podem ser vistos ao tocar no componente, como mostrado na Fig. 5. Mantendo o componente premido no modo de desempenho permite ao utilizador alterar a sua latência. As dicas de dados são também mostradas da mesma maneira que na versão para PC e, como são os últimos componentes da interface gráfica adicionados, são sempre mostradas no topo.

Os ficheiros de CPU, de instruções e de código são armazenados no directório de dados da aplicação, preferencialmente na memória externa. Esta memória, normalmente montada em `/mnt/sdcard`, é na verdade, em muitos dispositivos, uma partição da memória interna do dispositivo e não o cartão de memória externo. Mas, caso seja e não esteja disponível, a aplicação armazenará os ficheiros no directório privado da aplicação na memória interna do dispositivo. Armazenar os ficheiros na memória externa permite que o utilizador tenha acesso aos mesmos utilizando outra aplicação, como um explorador de ficheiros. Para saber o caminho para o ficheiro de CPU ou de código actualmente usado, o utilizador pode tocar no nome do ficheiro na interface gráfica.

V. CONCLUSÕES

Uma ferramenta para apoiar estudantes e professores no ensino e aprendizagem de arquitectura de computadores foi apresentada. Esta ferramenta, um simulador do MIPS, é versátil, intuitiva, configurável e agrega várias funcionalidades que se encontram dispersas por várias ferramentas existentes. Ambas as versões, uniciclo e *pipelined*, do CPU são suportadas, o caminho de dados é apresentado graficamente e um "modo de desempenho" está disponível. Esta ferramenta, o DrMIPS, está também disponível para Android, sendo esta uma inovação que a distingue de outras ferramentas semelhantes.

O DrMIPS está actualmente a ser utilizado na unidade curricular de Arquitectura e Organização de Computadores na FEUP [20]. O simulador é um programa gratuito, livre e de código fonte aberto.Pode ser descarregado de [21], e também se encontra disponível nos repositórios oficiais de duas distribuições de Linux, Debian e Ubuntu.

REFERÊNCIAS

[1] D. A. Patterson e J. L. Hennessy, "Computer Organization and Design - The Hardware/Software Interface", 3rd ed. Morgan Kaufmann, 2005.

[2] J. L. S. C. Pereira, "Educational package based on the MIPS architecture for FPGA platforms", Master Thesis, Faculdade de Engenharia da Universidade do Porto, Junho 2009, acedido em 19 de Novembro de 2014, http://repositorio-aberto.up.pt/bitstream/10216/59975/1/000135086.pdf.

[3] J. Larus, "SPIM: A MIPS32 Simulator",acedido em 19 de Novembro de 2014, http://spimsimulator.sourceforge.net.

[4] D. K. Vollmar e D. P. Sanderson,"MARS: An Education-Oriented MIPS Assembly Language Simulator", Março 2006, acedido em19de Novembro de 2014, http://www.cs.missouristate.edu/~vollmar/MARS/fp288-vollmar.pdf.

[5] J. Larus, "SPIM S20: A MIPS R2000 Simulator", Computer Sciences Department, Universityof Wisconsin, Tech. Rep., 1990, acedido em19 de Novembro de 2014, http://phoenix.goucher.edu/~kelliher/f2005/cs220/spim.pdf.

[6] G. C. R. Sales, M. R. D. Araújo, F. L. C. Pádua, e F. L. C. Júnior, “MIPS X-Ray: A Plug-in to MARS Simulator for Datapath Visualization”, 2010.

[7] J. Garton, “ProcessorSim - A Visual MIPS R2000 Processor Simulator”, 2005, acedido em 19 de Novembro de 2014, http://jamesgart.com/procsim.

[8] H. Sarjoughian, Y. Chen, e K. Burger, “A Component-based Visual Simulator for MIPS32 Processors”, 38th Frontiers in Education Conference, pp. F3B-9 - F3B-14, Outubro 2008.

[9] A. Gascoyne-Cecil, “MIPS-Datapath”,acedido em 19 de Novembro de 2014, http://mi.eng.cam.ac.uk/~ahg/MIPS-Datapath.

[10] I. Branovic, R. Giorgi, e E. Martinelli, “WebMIPS: A New Web-Based MIPS Simulation Environment for Computer Architecture Education”, Workshop on Computer Architecture Education, 31st International Symposium on Computer Architecture, 2004, acedido em 19 de Novembro de 2014, http://www4.ncsu.edu/~efg/wcae/2004/submissions/giorgi.pdf.

[11] T. E. Team, “EduMIPS64”, acedido em 19 de Novembro de 2014, http://www.edumips.org.

[12] D. Patti, A. Spadaccini, M. Palesi, F. Fazzino, e V. Catania, “Supporting Undergraduate Computer Architecture Students Using a Visual MIPS64 CPU Simulator”, IEEE Transactions on Education, vol. 55, no. 3, pp. 406 - 411, Agosto 2012, acedido em 19 de Novembro de 2014.

[13] “EduMIPS64 Students Questionnaire”,acedido em 19 de Novembro de 2014, http://www.diit.unict.it/users/spadaccini/edumips64-survey.html.

[14] M. Scott, “WinMIPS64”, Abril 2012, acedido em 19 de Novembro de 2014, http://indigo.ie/~mscott.

[15] Jimmat, “Assembly Emulator - Android Apps on Google Play”, Maio 2013, acedido em 18 de Novembro de 2014, http://play.google.com/store/apps/details?id=gr.ntua.ece.assembly.emulator.

[16] M. Butler, “Android: Changing the Mobile Landscape”, IEEE Pervasive Computing, vol. 10, no. 1, pp. 4 - 7, Janeiro-Março 2011.

[17] R. Shim, “Tablets Impact the Notebook Market: Enter the Ultrabook”, Information Display, vol. 28, no. 2 and 3, pp. 12 - 14, Fevereiro/Março 2012, acedido em 19 de Novembro de 2014, http://informationdisplay.org/Portals/InformationDisplay/IssuePDF/03_2012.pdf#page=14.

[18] Google, “Android SDK | Android Developers”, acedido em 27 de Novembro de 2014,http://developer.android.com/sdk/index.html.

[19] Fifesoft, “RSyntaxTextArea – Fifesoft”, 2013, acedido em 19 de Novembro de 2014, http://fifesoft.com/rsyntaxtextarea.

[20] Faculdade de Engenharia da Universidade doPorto, “FEUP - Arquitectura eOrganização de Computadores”, 2014, acedido em 27deNovembro de 2014, http://sigarra.up.pt/feup/pt/ucurr_geral.ficha_uc_view?pv_ocorrencia_id=350484.

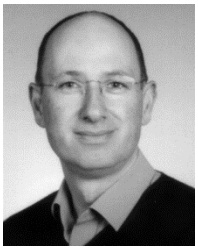
[21] B. Nova, “DrMIPS Wiki - Bitbucket”, Novembro 2014, https://bitbucket.org/brunonova/drmips.

AGRADECIMENTOS

O autor gostaria de agradecer ao Departamento de Engenharia Informática da FEUP e aos professores António Araújo e João Canas Ferreira do Departamento de Engenharia Electrotécnica e de Computadores pela orientação e conselhos dados, e também por propor este trabalho.

Bruno Nova nasceu no Porto, Portugal, em 1990, e concluiu um Mestrado Integrado em Engenharia Informática e Computação na Faculdade de Engenharia da Universidade do Porto em 2013.É o autor da Dissertação de Mestrado intitulada Tool to Support Computer Architecture Teaching and Learning, e de um artigo com o mesmo nome apresentado na conferência CISPEE 2013.

António Araújo obteve o grau de Doutor em Engenharia Electrotécnica e de Computadores na Faculdade de Engenharia da Universidade do Porto, Portugal, em 2003. É Professor Auxiliar no Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto desde 2003. Os seus principais interesses de investigação incluem aritmética em vírgula flutuante (operadores e funções matemáticas elementares) ao nível dos algoritmos, arquitecturas, metodologias de projecto e implementação em *hardware* dedicado, e desenvolvimento de arquitecturas e sistemas digitais reconfiguráveis para aplicações específicas.É membro da Ordem dos Engenheiros (Portugal).



João C. Ferreira obteve o grau de Doutor em Engenharia Electrotécnica e de Computadores (Universidade do Porto) em 2001. Actualmente é Professor Auxiliar no Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto. As suas actividades de investigação abrangem os sistemas embarcados reconfiguráveis dinamicamente (em FPGA), arquitecturas de sistemas digitais dedicados de alto desempenho, e ferramentas de apoio ao projecto de sistemas digitais. É membro da Ordem dos Engenheiros (Portugal), do IEEE e da ACM.