Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e Computadores
Projeto e Seminário
Ano letivo 2015/2016

PDS16inEcplise

Proposta de projeto

Estudantes:
André Ramanlal, 39204, 917539700, 39204@alunos.isel.ipl.pt
Tiago Oliveira, 40653, 938753297, 40653@alunos.isel.ipl.pt

Orientadores:
Tiago Dias (ISEL), tdias@cc.isel.ipl.pt
Pedro Sampaio (ISEL), psampaio@cc.isel.ipl.pt

1. Enquadramento

No domínio da Informática, um programa consiste no conjunto das instruções que define o algoritmo desenvolvido para resolver um dado problema usando um sistema computacional. Para que esse sistema possa realizar as operações definidas por estas instruções é pois necessário que as mesmas sejam descritas usando a linguagem entendida pela máquina, que consistem num conjunto de bits com valores lógicos diversos. Esta forma de codificação de algoritmos é bastante complexa e morosa, pelo que o processo habitual de desenvolvimento de um programa é feito com um maior nível de abstração, recorrendo a linguagens de programação. A Figura 1 mostra as diferentes fases deste processo quando aplicado ao domínio dos sistemas embebidos, em que as linguagens de programação mais utilizadas são o C e o C++.

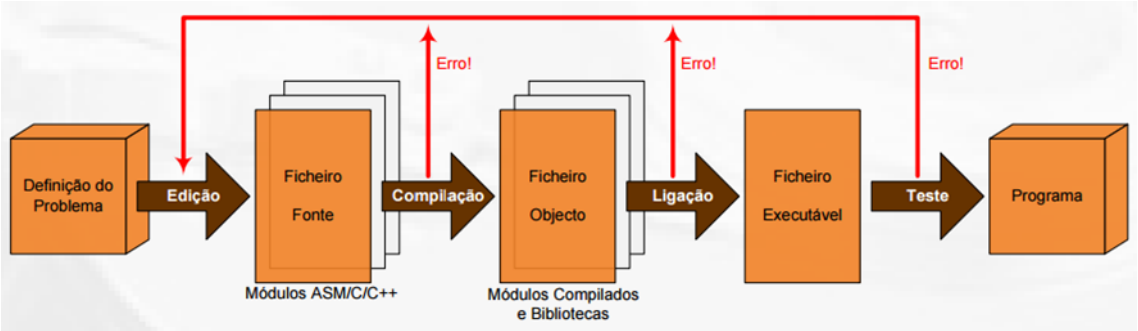


Figura 1 - Exemplo de um ciclo de desenvolvimento de um programa/aplicação. [1]

Após a definição do problema e elaboração do algoritmo para a sua solução, o programador começa a escrever o programa numa dada linguagem, resultando assim num ou vários ficheiros fonte. Estes são de seguida compilados através de um compilador ou *assembler*, que primeiramente verificam as regras sintáticas e

semântica da linguagem e de seguida geram um ficheiro objeto correspondente a cada ficheiro fonte. O *linker* efetua a ligação entre os diversos ficheiros objeto que compõem o programa e as bibliotecas utilizadas, ficheiros partilháveis que podem conter código, dados e recursos em qualquer combinação. Deste último processo resulta um ficheiro com a descrição do algoritmo codificado pelos programadores em linguagem máquina, i.e. um ficheiro executável. Para garantir a correta implementação da solução desejada, são realizados um conjunto de testes sobre este ficheiro.

Os Ambientes Integrados de Desenvolvimento (*IDEs*) são hoje em dia um enorme apoio no desenvolvimento destes programas, uma vez que não só disponibilizam diversas ferramentas para apoio à produção do código, e.g. um editor de texto, a geração automática de código ou o *refactoring*, como ainda possibilitam a interação com outras ferramentas e aplicações, como é o caso dos compiladores, *linkers*, *debuggers*, controladores de versão, etc.

Recorrendo a estas aplicações, um programador consegue ver a sua produtividade maximizada nas diferentes fases do processo de geração do ficheiro executável correspondente ao seu programa. Por exemplo, a geração automática de código permite poupar bastante tempo na escrita do código fonte do programa, bem como ter o código sempre bem indentado e estruturado. Já a funcionalidade de *syntax highlighting* facilita a leitura e análise do código fonte, para além de potenciar a deteção de erros de sintaxe e/ou de semântica. A utilização de um compilador integrado no IDE também permite acelerar o processo de geração do ficheiro executável, pois evita a saída do editor, a subsequente instanciação do compilador num processo aparte e, caso a compilação seja abortada devido a erros, a procura da linha associada a esse erro novamente no editor com vista à sua correção.

Atualmente, existem IDEs para quase todas as linguagens de programação em uso. Algumas destas aplicações suportam apenas uma linguagem de programação, como por exemplo o Kantharos que apenas suporta PHP ou o DRJava que apenas suporta Java. Não obstante, há vários IDEs no mercado que permitem desenvolver programas e aplicações usando várias linguagens de programação, tais como o Eclipse [2] e o IntelliJ [3] cuja quota de mercado é, à data atual, superior a 80%[4]. Esta versatilidade é normalmente conseguida à custa da adição de *plug-ins* ou *add-ons* específicos para uma dada linguagem de programação ao IDE, que são programas que ajudam adicionar novas funcionalidades aos *plug-ins*. Estes podem ser criados a partir de bibliotecas que dão o suporte à criação dos mesmos.

Apesar da maioria destes IDEs e dos seus *plug-ins* e *add-ons* estarem normalmente associados ao desenvolvimento de programas utilizando linguagens de alto nível, como é o caso do C, C++, C# ou Java, muitas destas aplicações também oferecem suporte à codificação dos programas, ou dos seus módulos, usando linguagens de mais baixo nível, tal como o *assembly* (e.g. o Eclipse).

2. Motivação

A arquitetura PDS16 [5] foi desenvolvida no Instituto Superior de Engenharia de Lisboa (ISEL), em 2008, com o objetivo de suportar não só uma mais fácil compreensão mas também o ensino experimental dos conceitos básicos subjacentes ao tema “Arquitetura de Computadores”. Esta arquitetura adota a mesma filosofia das máquinas do tipo *Reduced Instruction Set Computer* (RISC), oferecendo o seu ISA (*Instruction Set Architecture*) ao programador 6 registos de uso geral e cerca de 40

instruções distintas, organizadas em três classes: 6 instruções para controlo do fluxo de execução, 18 instruções de processamento de dados e 12 instruções de transferência de dados. O espaço de memória útil, que é partilhado para o armazenamento do código e dos dados dos programas, é endereçável ao byte e tem uma dimensão total de 64 kB.

Atualmente, o desenvolvimento de programas para esta arquitetura pode ser feito utilizando a própria linguagem máquina ou *assembly*. A tradução do código *assembly* para linguagem máquina é realizada recorrendo à aplicação dasm [6], que consiste num *assembler* de linha de comandos que apenas pode ser executado em sistemas compatíveis com o sistema operativo Windows da Microsoft.

Assim, o ciclo de geração de um programa passa por codificá-lo em linguagem *assembly* utilizando um editor de texto simples, tal como o Notepad, e posteriormente invocar a aplicação dasm a partir de uma janela de linha de comandos. Sempre que existam erros no processo de compilação, é necessário voltar ao editor de texto para corrigir a descrição *assembly* do programa e invocar novamente o *assembler*.

3. Objetivos

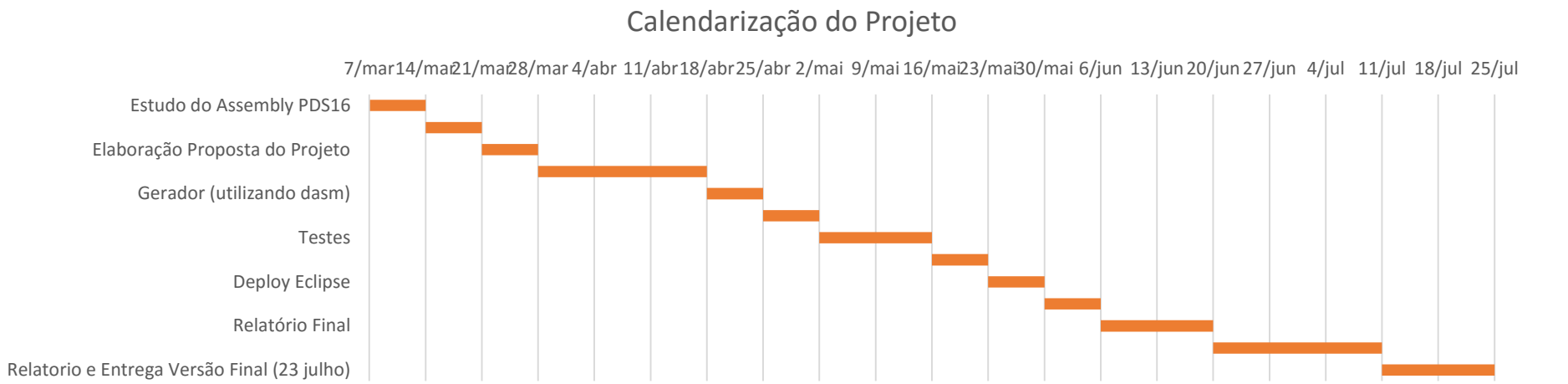
Com este trabalho pretende-se implementar um IDE para suportar o desenvolvimento de programas para o processador PDS16 usando a linguagem *assembly* e com as seguintes ferramentas e funcionalidades:

- Um editor de texto que integre ferramentas para fazer uma verificação da semântica e da sintaxe em tempo de escrita de código, de modo a que o programador possa ser alertado de eventuais erros na utilização da linguagem mais cedo e dessa forma otimizar a sua produtividade;
- *Syntax highlighting*, para permitir uma melhor legibilidade do código fonte;
- Integração com um *assembler*, para permitir a compilação dos programas sem necessidade de ter que abandonar o IDE e visualizar no editor de texto os eventuais erros detetados neste processo.

O IDE a desenvolver será baseado na plataforma Eclipse, atendendo à sua maior utilização na produção de programas e aplicações no domínio dos sistemas embebidos [7], onde se insere a utilização da arquitetura PDS16 no ISEL, e no facto dos alunos dos cursos de LEIC e LEETC do ISEL já terem experiência na utilização desta plataforma quando iniciam a frequência da unidade curricular Arquitetura de Computadores.

Para tal, será desenvolvido um *plug-in* para a arquitetura PDS16 utilizado a *framework* Xtext [8], que é uma *framework* genérica para o desenvolvimento de linguagens específicas de domínio (*DSL*). Para além da sua grande atualidade, a *framework* Xtext apresenta ainda a grande vantagem de, com base numa mesma descrição de uma DSL, permitir gerar automaticamente *plug-ins* também para a plataforma IntelliJ e para vários *browsers.*

4. Calendarização



	Relatorio e Entrega Versão Final (23 julho)	"Tab" de opção para o compilador - Opcional	Relatório Final	Deploy IntelliJ	Deploy Eclipse	Cartaz e Versão beta (13 Junho)	Testes	Relatório de Progresso	Gerador (utilizando dasm)	Implementação do ASM PDS16	Elaboração Proposta do Projeto	Estudo da Framework Xtext	Estudo do Assembly PDS16
Start Date	11/jul	20/jun	6/jun	30/mai	23/mai	16/mai	2/mai	25/abr	18/abr	28/mar	21/mar	14/mar	7/mar
Days to Complete	14	21	14	7	7	7	14	7	7	21	7	7	7

Figura 2- Diagrama de Gantt relativo à previsão da execução do trabalho.

5. Referências

[1] Dias, Tiago (2013). Elaboração de Ficheiros Executáveis. Obtido em 2016/03/27 de https://adeetc.thothapp.com/classes/SE1/1314i/LI51D-LT51D-MI1D/resources/2334

[2] IDE Eclipse, http://www.eclipse.org

[3] IDE Intellij, https://www.jetbrains.com/idea/

[4] White, Oliver (2014). IDEs vs. Build Tools: How Eclipse, IntelliJ IDEA & NetBeans users work with Maven, Ant, SBT & Gradle. http://zeroturnaround.com/rebellabs/ides-vs-build-tools-how-eclipse-intellij-idea-netbeans-users-work-with-maven-ant-sbt-gradle/. Consultado a 2016/03/25

[5] Paraíso, J. (2011). PDS16. Arquitetura de Computadores – Textos de apoio às aulas teóricas (págs. 13-1 – 13-27), Lisboa.

[6] Paraíso, J. (2011). Desenvolvimento de Aplicações. Arquitetura de Computadores – Textos de apoio às aulas teóricas (págs. 15-2 – 15-5), Lisboa.

[7] Cherly Ajluni. Eclipse Takes a Stand for Embedded Systems Developers, http://www.embeddedintel.com/search_results.php?article=142. Consultado a 2016/03/30.

[8] (2013). Xtext 2.5 Documentation, Eclipse Foundation. Obtido em 2016/02/05 de http://www.eclipse.org/Xtext/documentation/2.5.0/Xtext%20Documentation.pdf