1º Trabalho Prático - Sistemas Embarcados
Especificação

Objetivo

Explorar as relações de custo-benefício nas diferentes arquiteturas e organizações disponíveis para processadores embarcados. Neste trabalho, consideraremos a arquitetura de propósito geral ARM, os processadores DSP da família Blackfin e os microcontroladores 8051.

Tarefas

1) Buscar os seguintes dados sobre as arquiteturas consideradas: custo (em dólares), frequência de operação, potência média dissipada.

2) Para a arquitetura ARM, modele da forma mais fiel possível os processadores escolhidos utilizando o simulador gem5. Para as demais arquiteturas, utilize os simuladores dedicados.

3) Desenvolver três algoritmos que ataquem, na sua concepção, três campos diferentes (e.g.: multiplicação de matrizes, Fast Fourier Transform e Quick Sort). Lembre-se: Tempo de execução = #ciclos / frequência. **Obs: os algoritmos devem ser diferentes dos supracitados, que estão disponíveis como exemplos na imagem disponibilizada.** Execute esses algoritmos nas arquiteturas modeladas.

4) Selecione quais das arquiteturas você colocaria em um dispositivo embarcado executando cada um dos algoritmos. Considere que existem dependências de dados entre os algoritmos, ou seja, eles devem ser executados um após o outro, mesmo que não seja no mesmo dispositivo. Considere um cenário onde queremos a melhor relação entre custo, em dólares, e desempenho, em segundos. Portanto, o objetivo é tentar minimizar o produto custo-desempenho (em US$.s) dado por: custo total dos dispositivos selecionados × tempo total de execução. Isto é, pode-se construir um cenário em que um microcontrolador e um DSP são melhores que um ARM apenas.

5) Durante a realização dos passos 1 e 4 você deve ter notado uma grande disparidade nas frequências de operação dos dispositivos utilizados. Isso se deve, entre outros motivos, pelas diferentes tecnologias de fabricação utilizadas. Para compararmos as diferentes arquiteturas relevando este fato, repita o passo 4 considerando paridade de frequência. Considere que todos os dispositivos utilizam uma frequência de 1GHz.

6) **Repita o passo 4**, agora considerando um cenário onde o objetivo é minimizar o consumo de energia. A energia consumida é dada por tempo de execução × potência média dissipada. Considere que somente o dispositivo que está executando consome energia e que os demais estão em um estado de baixo consumo, no caso de sistemas que teriam múltiplos processadores.

Organização dos Grupos e Método de Apresentação

O trabalho deve ser realizado em grupos de dois ou três alunos. O produto final do trabalho será uma apresentação em Power Point. Apresente brevemente os modelos de processadores considerados e a modelagem feita para os processadores ARM. Inclua também uma tabela com os tempos de execução obtidos. Apresente as escolhas feitas para os passos 4 a 6 e os valores obtidos para os produtos custo-desempenho e energia consumida. Para os passos 4 a 6 discuta também cenários reais onde cada uma das métricas objetivo se aplica.

Importante: Não se esqueça de incluir, ao final do documento, todas as fontes utilizadas para obter informações e valores dos componentes considerados.

Método de Avaliação

O trabalho será apresentado para toda a turma por cada grupo (as apresentações serão divididas em dois dias). A nota de cada grupo será definida pela nota do professor e pela nota dadas aos alunos. Os alunos deverão anotar a nota de cada grupo e posteriormente enviar pelo Moodle, em um link que ficará disponível para isto.

O que deverá ser considerado:

30% - Qualidade da apresentação (e.g..: o quão clara está a apresentação; como o apresentador interage com a apresentação; como os dados estão sendo mostrados)

30% - Pensamento crítico (domínio do grupo em formular teorias para discutir os seus resultados)

40% - Complexidade (qual a dificuldade dos experimentos que o grupo fez? Um maneira prática de mensurar isto é fazer a seguinte pergunta: “este grupo teve mais trabalho do que o nosso?”; ou “pensou melhor no trabalho que o nosso?”).

Documento de Auxílio

Os links abaixo contém duas máquinas virtuais com os simuladores necessários para este trabalho pré-instalados. Uma VM contém os simuladores VisualDSP++ e RKit, e a outra VM contém o Gem5.

VM Windows com VisualDSP++ e RKit
VM Ubuntu com Gem5

Faça download do VirtualBox aqui:
https://www.virtualbox.org/

Importe cada máquina virtual no VirtualBox e execute-a.

A seguir são apresentados os simuladores e algumas dicas úteis necessárias para a simulação dos programas.

GEM5

O GEM5 é uma plataforma de simulação de sistemas computacionais. Ela permite organizar e parametrizar os componentes destas arquiteturas, como por exemplo: definição do processador (ARM, MIPS, Intel X86, etc); tamanho da memória cache e memória principal; tipo de CPU (In Order, Out-of-Order, Atomic, etc), dentre outras. Ele também permite simular a execução de sistemas operacionais e aplicações, possibilitando verificar o andamento de sua execução em tempo real. Quando uma simulação é finalizada, ela fornece ao desenvolvedor dados suficientes para analisar o comportamento das aplicações executadas sobre tais arquiteturas, como por exemplo: número total de acessos à memória cache e principal, total de instruções executadas, etc. Informações completas sobre este simulador podem ser encontradas neste link: www.m5sim.org/Main_Page.

O GEM5 será utilizado para simular os processadores ARM. Abaixo é descrito um mini-tutorial em etapas sobre sua utilização nesta disciplina.

1ª Etapa – Acesso: O GEM5 está instalado na VM disponibilizada.

2ª Etapa – Simulação

Descompactar o arquivo .ova, que contém a imagem da máquina virtual. Para abrir o arquivo, pode-se utilizar qualquer programa compatível com o formato (VirtualBox, por exemplo, é um programa gratuito e multiplataforma que pode ser utilizado). Uma vez que o programa (ex. VirtualBox) esteja instalado, importar o arquivo .ova disponibilizado.

A senha de acesso é *orgb*. Assim como o nome de usuário.

Comandos no terminal do linux indicados a seguir são precedidos de '$', exemplo:
$ comando_qualquer

Depois de logar no sistema linux pela máquina virtual, abrir o terminal, pela barra de aplicativos à esquerda, atalho na área de trabalho ou atalho Ctrl + Alt + T. Para colar comandos exemplo no terminal utilize o atalho Ctrl + Shift + V. Se você quiser copiar e colar texto entre a sua máquina *host* (seu computador) e a máquina *guest* (sistema na máquina virtual) pode ser necessário instalar complementos da máquina virtual.

Acesse, via terminal, a pasta do gem5 com o comando Change Directory (cd):
$ cd gem5

Para listar os arquivos e diretórios, use o comando List (ls)
$ ls

Execute um programa exemplo (hello world) no gem5. Além do executável do gem5, é preciso especificar um arquivo de configuração de simulação em python (use o arquivo *st_sim.py* localizado na pasta emb). A flag '-c' é o nome do programa a ser executado. Se for necessário passar algum argumento para o programa, ainda pode-se adicionar a flag '-o' seguido da lista de argumentos separados por espaços. Por exemplo, o comando abaixo executa um hello world, já compilado para ARM, no gem5 simulando um processador ARM A15:

$ build/ARM/gem5.opt emb/st_sim.py -c tests/test-progs/hello/bin/arm/linux/hello --cpu ARM_A15

Se a simulação estiver correta, um novo arquivo arquivo 'stats.txt' será gerado na pasta 'm5out', dentro da pasta do gem5. Você pode abrir o arquivo pela interface gráfica do Ubuntu,

abrindo o navegador de pastas que também está na barra de aplicativos à esquerda. No arquivo config.ini é possível visualizar as configurações do processador que foram usadas na simulação.

A flag --cpu especifica o processador simulado. Neste exemplo (ARM_A15) as configurações usadas na simulação são as definidas no arquivo ARM_A15.py localizado na pasta "emb". Nesta pasta é possível encontrar outros exemplos de configuração de cpu. **O aluno deverá modificar os parâmetros do processador e cache em um destes arquivos para que o sistema simulado seja o mais similar possível com o processador escolhido.**

Seus benchmarks devem ser compilados estaticamente para executarem no gem5. Uma pasta modelo para programas já está criada dentro da pasta do gem5. Acesse-a e compile o programa hello.c, exemplo:

$ cd emb_exemplo
$ arm-linux-gnueabi-gcc -static hello.c -o hello

A flag -o indica o nome de saída do executável, e a flag -static indica que as bibliotecas necessárias serão estaticamente linkadas.

Volte para a pasta base do gem5 testar o programa:
$ cd ..
".." indica o diretório pai.

Simule no gem5 o arquivo que você acabou de compilar (conforme instrução abaixo). Após a execução um novo arquivo stats.txt estará disponível na pasta m5out.

$ build/ARM/gem5.opt emb/st_sim.py -c emb_exemplo/hello *--cpu ARM_A15*

O procedimento é semelhante para os benchmarks escolhidos para utilizar no trabalho: Compile e simule.

OBS1: Seu benchmark deve ser escrito na linguagem C.

OBS2: A execução de um benchmark em um simulador é diversas vezes mais lento do que em um processador real. Portanto, o aluno deve mensurar bem o tamanho das entradas do benchmark de forma que o simulador consiga concluir a aplicação e que os resultados ainda sejam significativos.

Documentos úteis:

http://pages.cs.wisc.edu/~markhill/cs757/Spring2012/includes/isca_pres_2011.pdf
http://www.m5sim.org/Tutorials
http://www.m5sim.org/Tutorials#HiPEAC_2012_Computer_Systems_Week

VisualDSP++

Utilizado para simulação de processadores DSP. Está disponível na imagem do Virtualbox disponibilizada.

Atenção: Antes de executar o simulador desative a sincronização do relógio entre a máquina virtual e a hospedeira (um tutorial com a primeira solução é fornecido aqui) OU atrase o relógio da máquina hospedeira. Ao abrir a VM, atrase o relógio da máquina virtual em alguns anos (configure a data para 19/09/2013), e desative a opção "Sincronizar automaticamente com um servidor de horário na Internet". Estes procedimentos são necessários para evitar problemas com a licença do software.

Na área de trabalho existe a pasta 'Trabalho 1 Simuladores'. Nesta pasta você encontrará os seguintes arquivos:

- Manual da ferramenta (Simulador DSP/guide.pdf)
- Atalho do programa (duplo clique para executar). A página inicial do programa:

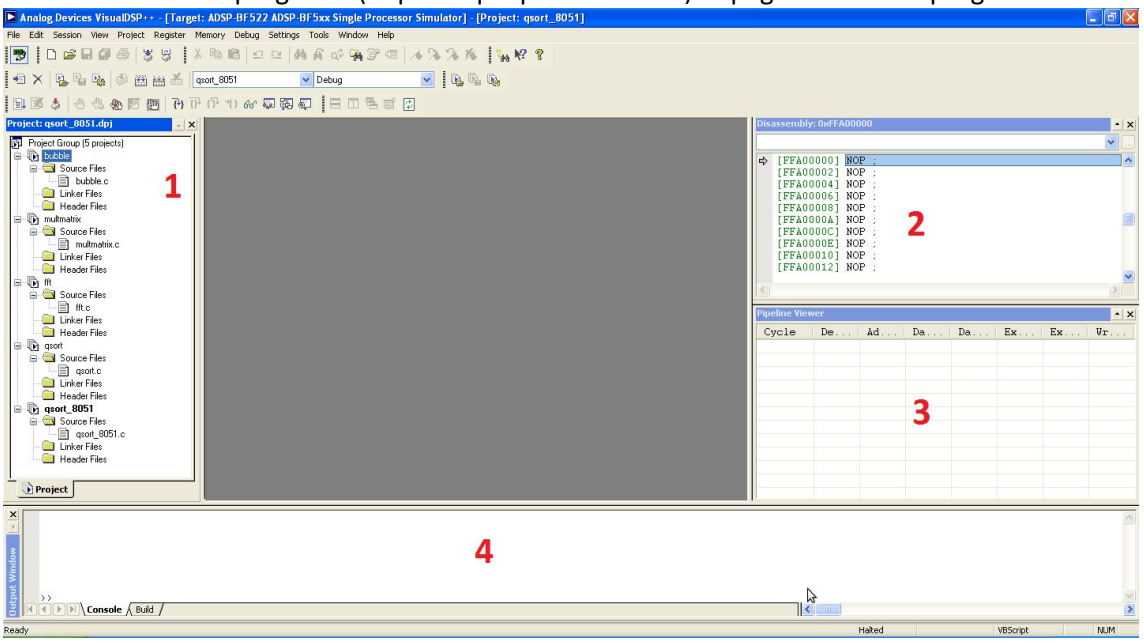
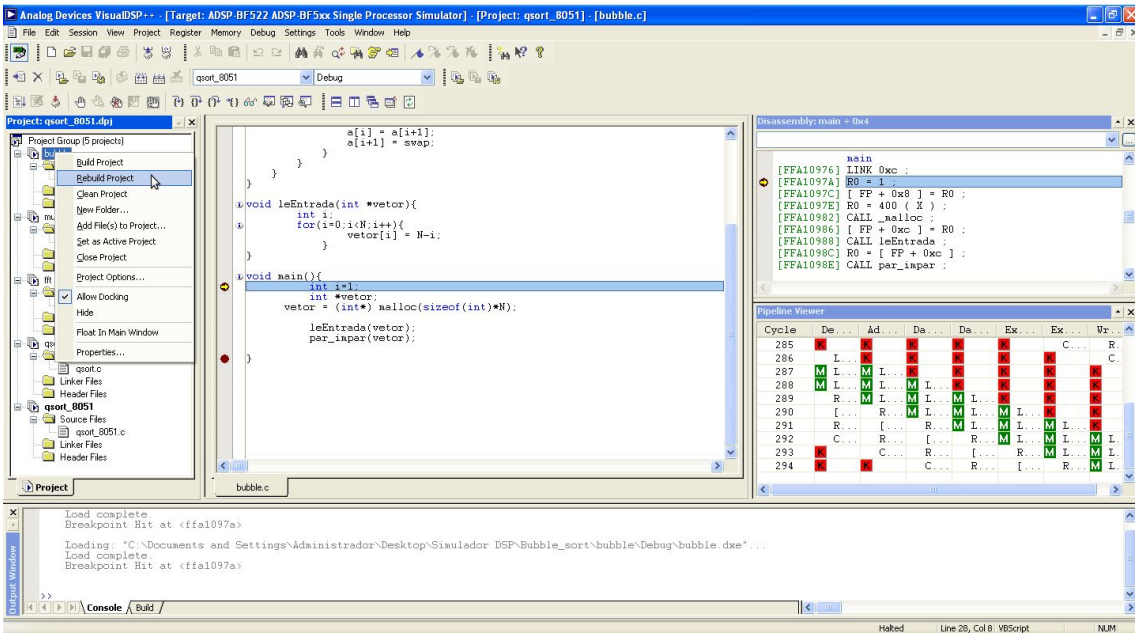


Fig. 1. Visualização inicial do simulador Visual DSP++

1. Visualização dos projetos. (Por padrão já possui para os algoritmos em questão).
2. Visualização do Disassembly (Aparecerá quando compilar algum código).
3. Visualização do pipeline durante a execução da aplicação
4. Console de saída. Onde aparecem informações sobre compilação (sucesso ou falha). Saídas do programa (printf...)

Para abrir um arquivo, de duplo clique sobre ele (neste exemplo, bubble.c no campo project). A Figura 2 (abaixo) apresenta a visualização do código fonte.



Ao meio da Figura 2 (acima) pode-se encontrar o código fonte, escrito em linguagem C. Para compilar o código – projeto, recomenda-se (por já ter dado problemas anteriores) clicar com o botão direito sobre o nome do projeto e em seguida clicar em “Rebuild Project”. A compilação será automática e já irá iniciar a seção de debug, que pode ser vista na Figura 3.

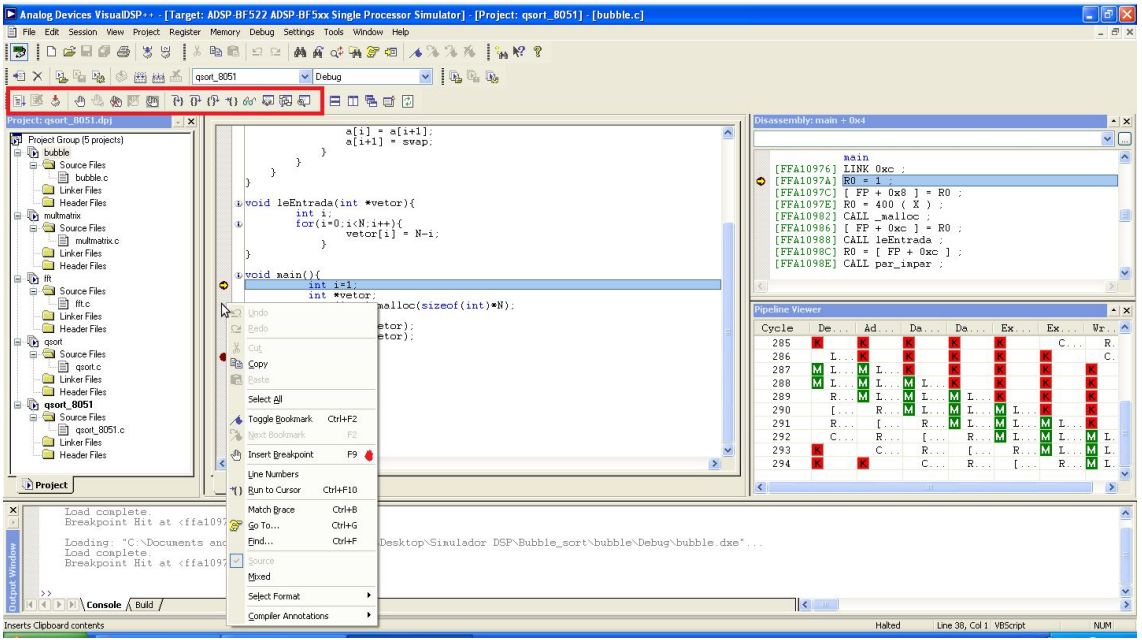


Fig. 3. Visualização durante seção de debug

A Figura 3 mostra a seção de debug. Recomenda-se a inserção de breakpoint ao final do código para coletar o número final de ciclos. Para inserir um breakpoint, posicionar sobre a linha desejada e clicar com o botão direito e após ir em Inserir Breakpoint.

Durante o debug, os botões internos ao retângulo destacado em vermelho irão auxiliar na simulação com funções de: “run”, “reset”, “step into”, step out”.
Ao finalizar a simulação, a quantidade de ciclos será exibida no “pipeline viewer”.

Dicas: Se der algum problema de travamento, finalizar o programa e executar novamente.

- Dependendo da aplicação que está simulando, poderá demorar maior tempo para simular.
- Quando o simulador estiver executando, ficará piscando no canto inferior direito com a mensagem: “running”. Ao final, será exibida a mensagem: “halted”.
- Caso ocorra erro em compilação (build) informando memory overflow, verificar o tamanho de entrada da aplicação.

RKit

Utilizado para simulação de Microcontroladores.

Está disponível na imagem do virtualbox disponibilizada. Na área de trabalho existe a pasta ‘Trabalho 1 - Simuladores’. Nesta pasta você encontrará os seguintes arquivos:

- Manuais da ferramenta (em ‘Simulador Microcontroladores’)
- Atalho do programa Ride7 (duplo clique para executar)
- Pastas com projetos de exemplo: Contém os projetos para os códigos já testados para o microcontrolador P80C51 e/ou C8051.

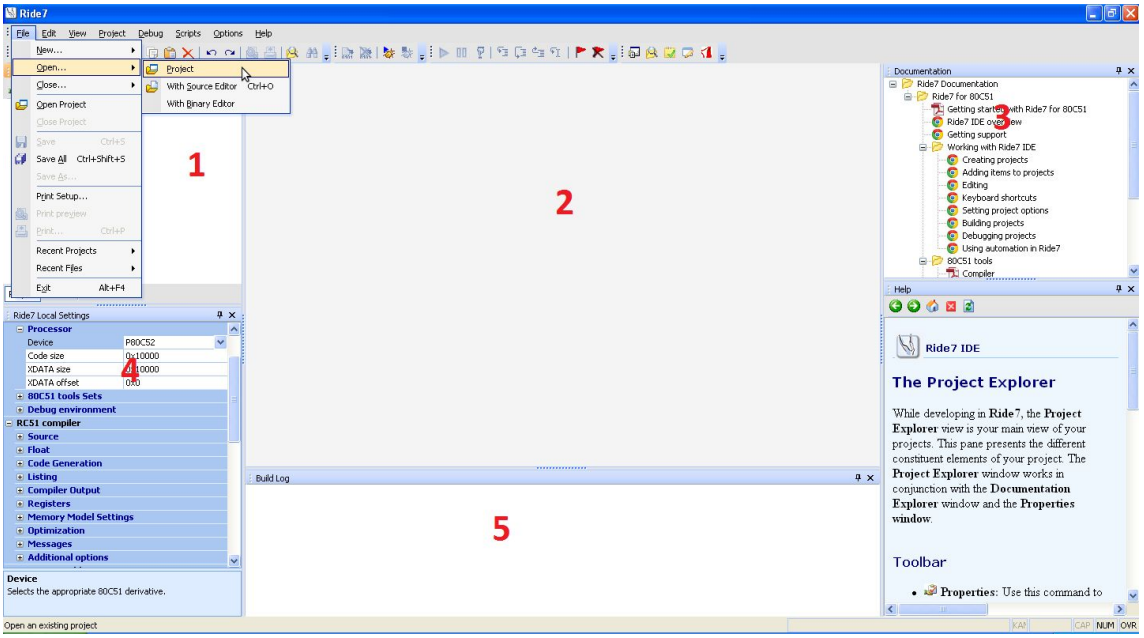


Fig. 4. Simulador Ride7

A Figura 4 mostra a tela inicial do simulador:

1 – Árvore do projeto, contendo os códigos fontes, bibliotecas...

2 – Campo onde será mostrado o código fonte em questão

3 – Documentação do processador/microcontrolador que está sendo simulado

4 – Descrição das características do processador/microcontrolador. Pode ser alterado conforme necessário.

5 – Campo de log da compilação e execução do debug.

Os projetos para os algoritmos em já estão criados. Para utilizá-los, vá em “File -> Open -> Project” e selecione um dos projetos. Os projetos existentes estão na pasta “Trabalho 1 Simuladores - Simulador Microcontroladores”. Selecionado o projeto, de duplo clique sobre ele e aguarde.

Neste exemplo, será aberto o projeto “mmult.prj”, que corresponde a multiplicação de matriz. Após o projeto estar carregado, de duplo clique sobre o código “mmult_8051.c” e ele será aberto para edição. A Figura 5 apresenta uma visão geral do projeto quando aberto.

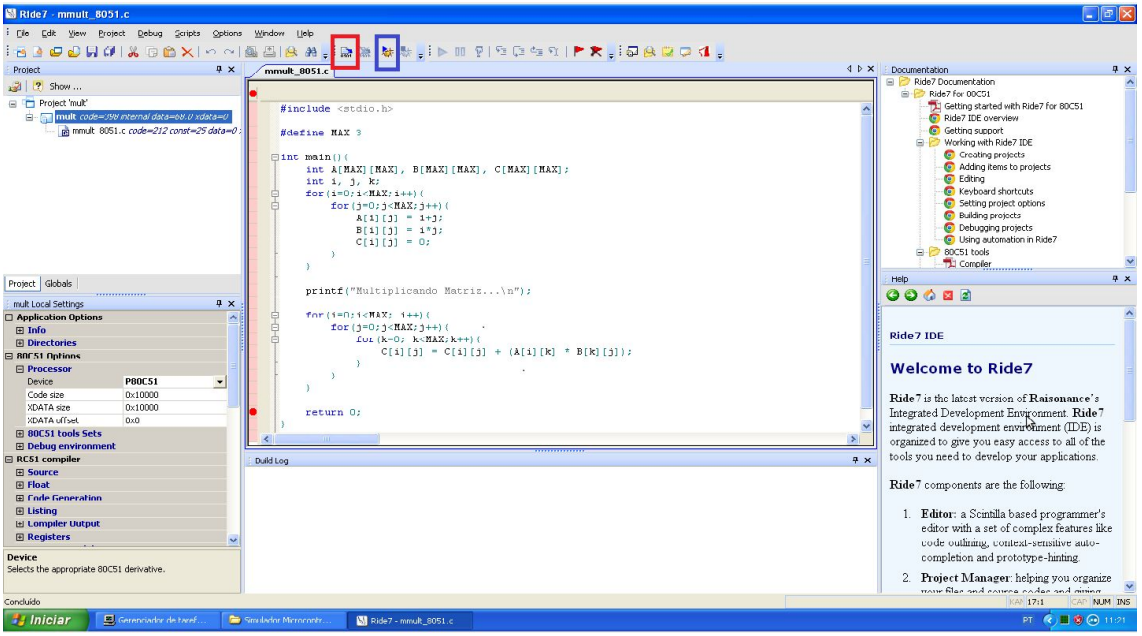


Fig. 5. Visão geral do projeto mmult.prj

Com o projeto em aberto, o próximo passo, se necessário, é editar o arquivo fonte. Para compilar ele para a arquitetura em questão, apertar o botão interno ao quadrado vermelho (destacado na Figura 5). Para iniciar a simulação (debug), clicar sobre o botão interno ao quadrado azul (destacado na Figura 5). O ambiente de debug é exibido na Figura 6.

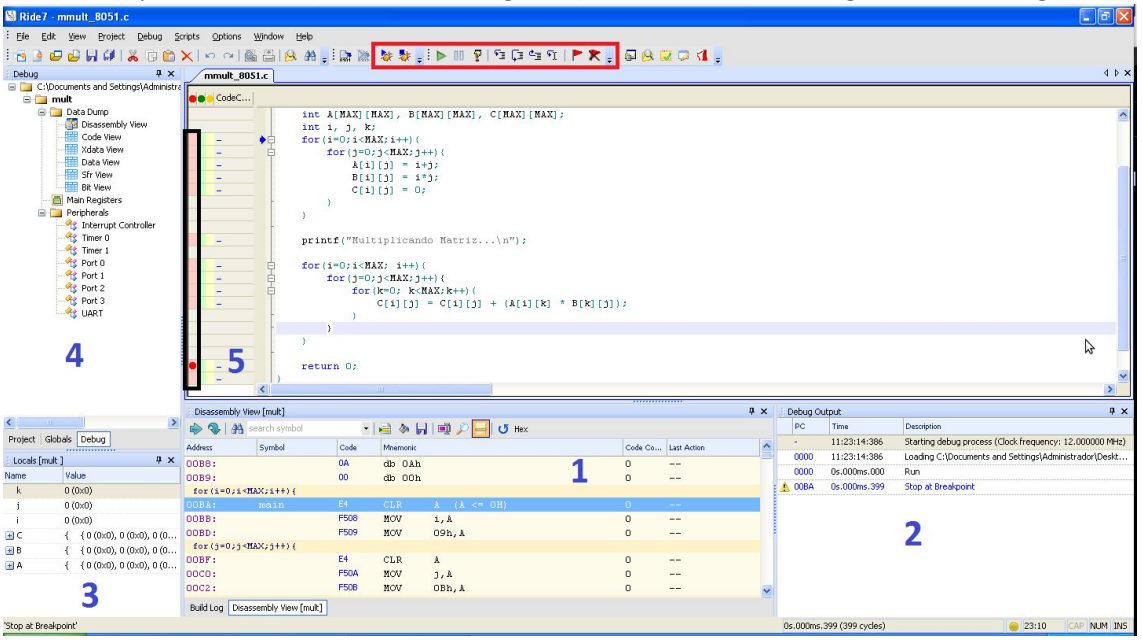


Fig. 6. Simulação (debug) da multiplicação de matriz

A seção de debug é composta, principalmente pelas seguintes ferramentas:

- As ferramentas internas ao retângulo vermelho (Figura 6), auxiliam na simulação (run, reset, step into, step out).

- O campo interno ao retângulo preto corresponde ao lugar de inserção de breakpoints. Para inserir um, basta clicar em cima do local desejado (em rosa).

- 1. Campo de exibição do disassembly (podendo ser visualizado em Hexadecimal).

- 2. Saída do Debug.

- 3. Locais (memória).

- 4. Debug do projeto. Pode-se acompanhar o funcionamento das portas (Port 0, 1, 2, 3) em tempo de execução. Para isto, clicar sobre o item.

- 5. Durante a simulação, ao invés dos “riscos azuis”, aparecerá a quantidade de vezes que foi executada a operação de tal linha.

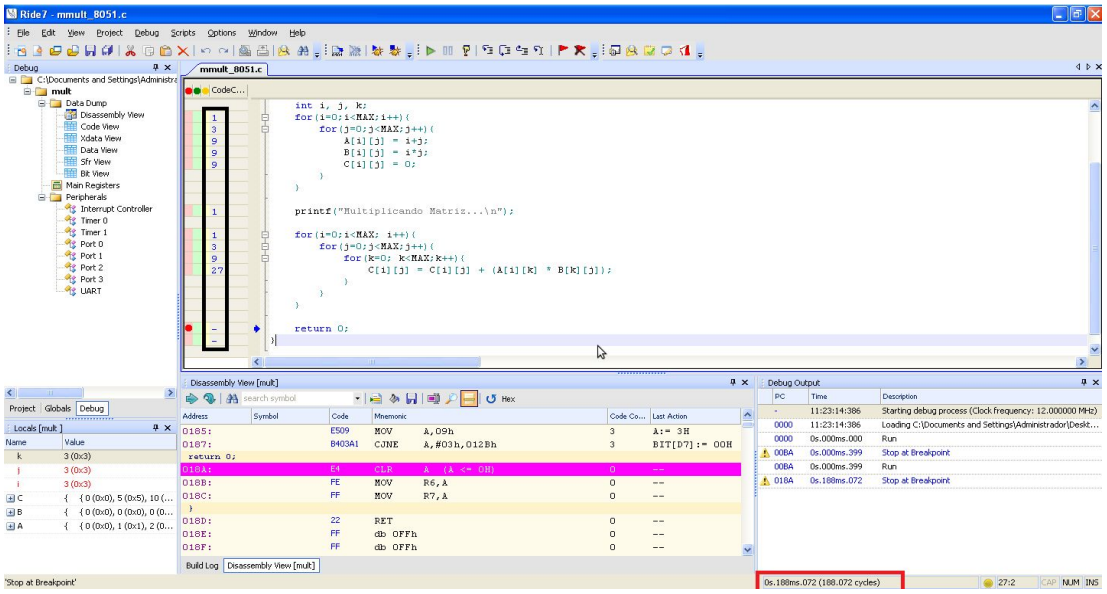


Fig. 7. Simulação encerrada

Antes de executar a simulação, recomenda-se inserir breakpoint ao final do código, conforme apresentado na Figura 6 (bola vermelha interna ao retângulo preto).

Ao final da simulação, será mostrada no canto inferior direito (interno ao retângulo vermelho) a quantidade de ciclos que o processador/microcontrolador levou para executar tal aplicação. Conforme mostrado na Figura 7. Já a quantidade de vezes que cada linha (operação) foi executada aparece interna ao retângulo preto.

A pasta “Trabalho 1 - Simuladores/códigos”, disponibilizada na Área de Trabalho contém alguns códigos de exemplo a serem utilizados, além dos que cada ferramenta possui de exemplos de seus projetos.

Algumas dicas

- O código de cada uma das aplicações de exemplo está disponível em uma versão para cada compilador. Utilize sempre a versão apropriada para cada ambiente. **Atenção:** verifique se o tamanho da entrada de cada programa é o mesmo para todas as arquiteturas.
- O simulador de 8051 não encerra a execução quando o programa é encerrado. Para medir o tempo que cada algoritmo leva, recomenda-se inserir um *breakpoint* (coluna vermelha à esquerda do código) na linha do comando *return* da função *main*.
- **Os ambientes de DSP e 8051 possuem diversos dispositivos diferentes. Recomenda-se iniciar a exploração do espaço de projeto com o DSP Blackfin BF533 e o microcontrolador AT89S51, da Atmel.**