Lista de Exercícios

1. Faça um texto, com uma lauda, sobre o Raspberry Pi, considerando os aspectos:

- Tamanho da palavra de memória
- Espaço de endereçamento de memória
- Número e uso de registradores
- Formato das instruções

2. Explique porque a arquitetura do Raspberry Pi é considerada uma arquitetura load/store. Quais são as principais vantagens e desvantagens desta arquitetura?

3. Faça um programa, em assembly, para ler dois números e indicar o maior.

4. Faça um programa, em assembly, calcular as raízes reais de uma equação do segundo grau, após ler os coeficientes através do teclado.

5. Faça um programa, em assembly, para calcular a combinação simples após ler a quantidade de elementos de um conjunto e o número de elementos que irão formar os agrupamentos.

Sugestão: visite http://mundoeducacao.bol.uol.com.br/matematica/combinacao-simples.htm

Os códigos fontes devem ser comentados, utilize o símbolo @.

Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso – Campus Cuiabá “Octayde Jorge da Silva”

Disciplina: Arquitetura e Organização de Computadores

Discente: Vitor Bruno de Oliveira Barth

Docente: Ed’Wilson Tavares Ferreira

Lista de Exercícios sobre Raspberry Pi e Assembly

Questão 1) Raspberry Pi é um computador de baixo custo mantido pela Raspberry Pi Foundation com a intenção de facilitar o ensino de computação. Hoje, em sua 3ª versão, ele conta com um System on Chip Broadcom BCM2837, que possui com um processador Cortex A53 ARM v8 com instruções de 64 Bits, quatro núcleos e clock de 1.2GHz e 1GB de memória RAM.

Porém a versão que utilizamos em aula utiliza uma simulação da versão anterior, o modelo 2, que possui um processador Cortex A7 ARM v6 de 900MHz Quad-Core, 512MB de RAM (reduzidas a 192MB devido a limitações do software de virtualização QEMU), e instruções de 32 Bits.

A arquitetura ARM tem um conjunto de instruções reduzido (RISC), no formato Load/Store, ou seja, para que sejam executadas operações na ALU é necessário que a informação tenha sido copiada para os registadores. Os dois processadores, Cortex A7 e Cortex A53 possuem 37 registradores, que armazenam uma palavra cada um, ou seja, 4 bytes no Cortex A7 e 8 bytes no Cortex A53.

Nem todos os registradores estão disponíveis para usuário. De fato, apenas 16 podem ser utilizados pelo usuário, sendo 13 de uso geral, um ponteiro de pilha (SP, Stack Pointer) e dois ponteiros de endereços (LR, Link Register, utilizado para retornar após uma subrotina, e PC, Program Counter, que indica a próxima instrução).

Assembly em arquiteturas ARM possuem poucas instruções (o que faz sentido, julgando que é uma arquitetura RISC), que são sempre escritas da seguinte forma:

INSTRUÇÃO{Cond} Rd, Rm, Rn, <Oprnd2>

Onde Rd costuma ser o Registradore de Destino, Rm e Rn os registradores onde estão salvo os operandos, e <Oprnd2> um valor ou um registrador. {Cond} é opcional, mas caso utilizado, faz com que essa instrução só seja realizada caso essa condição tenha sido atingida. Arquiteturas ARM v4 em diante suportam operações com números longos, ou seja, números que utilizam dois registradores, e então seria necessário mais um registrador de destino.

Questão 2) Para que o processador do Raspberry Pi realize operações na ALU (Unidade Lógica Aritmética) é necessário que eles estejam salvos em um registrador, ou seja, num circuito de memória integrado ao processador. Contudo, estes registradores são poucos e de pequeno tamanho, logo faz-se necessário o uso de uma memória externa. Buscando carregar (*load)* e guardar (*store)* informações na memória externa, é necessário executar uma instrução que realiza tais tarefas antes de se processar o dado. No processador com arquitetura ARM presente no *Raspberry Pi*, a instrução que carrega da memória é LDR e a que armazena em memória é STR.

A principal vantagem da arquitetura *load-store* é o reduzido ciclo de *clock* devido às instruções mais simples. Contudo, instruções mais simples exigem um trabalho maior por parte do desenvolvedor, pois torna necessário maior quantidade de operações para realizar uma tarefa. Esta arquitetura exige também que o programa administre diretamente os registradores, portanto quanto mais registradores estão disponíveis, mais difícil é administração deles. Porém, como não é possível ler e processar dados diretamente da memória, se consumirá muito tempo com operações *load-store* caso não haja registradores suficientes para a aplicação.

Questão 3, 4 e 5) Estão anexos em arquivos .zip. Colar o código aqui causaria confusão e um arquivo muito extenso, pois foram utilizados vários arquivos .s com centenas de linhas cada. Para compilar use o comando make que gerará um arquivo executável. A entrada e os resultados serão exibidos em tela, não sendo necessário verificar o valor de saída de R0, que nada significará.