IFES – Campus Serra
Engenharia de Controle e Automação
Arquitetura de Computadores
Professor: Rafael Emerick

Digitalize sua resolução manuscrita e envie o documento digital em PDF para contabilização de nota em atividade de EAD.

Lista de Exercícios 1

1) Considerando uma palavra de 4 bits e a o número 14(decimal). Mostre os resultado das operações 14<<2, 14>>2 e 14>>>2.

2) Considerando a seguinte operação aritmética: 12-7. Mostre a representação binária dessa operação utilizando notação sinal magnitude e complemento de dois. Quais as vantagens da utilização da representação em complemento de dois?

3) Explique a relação lógica entre Microarquitetura, Arquitetura, Sistema Operacional e Aplicação em software, quais as funções de cada elemento e quais os aspectos relevantes de cada uma dessas abstrações?

4) Faça um programa em Assembly ARM que calcule a sequência de Fibonacci até um determinado limite. A sequência deve ser registrada na memória em uma estrutura de vetor de números, e o valor limite deve ser lido de um endereço na memória. Escolha dois endereços adequados para registro da sequência e do valor limite. Crie o programa utilizando corretamente a segmentação de memória do ARM, com no mínimo, os segmentos .global e .text.

a) Implemente o código assembly ARM no simulador https://cpulator.01xz.net/?sys=arm. Valide a implementação acompanhando a ocupação da memória.

b) Escolha 2 instruções de Processamento de Dados, 2 instruções de acesso a memória e 2 de branch, calcule manualmente a conversão da instrução para código de máquina, e compare o resultado com o apresentado pelo opcode do simulador.

5) Considere uma arquitetura com palavra de 32 bits. Seja uma variável nessa arquitetura armazenada na 42º palavra de memória. Se a memória é endereçada a byte, responda:

a) Qual é o endereço de memória da 42ª palavra na memória.

b) Qual a faixa de endereço de memória que a palavra ocupará?

c) Desenhe o número 0xFF223344 armazenado na palavra 42 em uma máquina big-endian e em uma little-endian. Indique claramente o endereço de byte de cada um dos bytes armazenados.

6) [6.5] Explique como o seguinte programa ARM pode ser usado para determinar se um computador é big-endian ou little-endian.

```
MOV  R0, #100
LDR  R1, =0xABCD876    ; R1 = 0xABCD876
STR  R1, [R0]
LDRB R2, [R0, #1]
```

7) [6.7] Escreva as seguintes **strings** usando codificação ASCII, escreve a resposta em hexadecimal.

a) Racha Cuca

b) Tripa Seca

8) Faça o disassembly dos seguintes código de máquina
a) 0xE5902000
b) 0xE282200A

9) Com relação ao processo confecção de software, explique
a) Explique a diferença entre Compilação e Montagem de um programa.
b) Considerando o processo de transformação de uma linguagem de alto nível e de baixo nível, explique o processo de compilação do código para linguagem de máquina.
c) Qual a diferença entre um software compilado e um software interpretado? Explique

10) Sobre a estrutura de memória em pilha, responda:
a) O que é uma pilha em um computador?
b) Explique a diferença entre uma fila e uma pilha?
c) Qual é a importância da pilha durante as chamadas de funções?
d) O que uma função não-folha deve guardar na pilha além dos registradores de dados?

11)Considere o seguinte trecho de código C:

```
// C code
void setArray(int num) {
  int i;
  int array[10];

  for (i = 0; i < 10; i = i + 1)
    array[i] = compare(num, i);
}
int compare(int a, int b) {
  if (sub(a, b) >= 0)
    return 1;
  else
    return 0;
}
int sub(int a, int b) {
  return a - b;
}
```

a) Implemente esse trecho de código em linguagem de montagem ARM. Use R4 para armazenar a variável i. Verifique o manuseio correto do stack pointer. O array é armazenado na pilha da função *setArray*.

b) Considere que *setArray* é a primeira função chamada. Desenhe o status da pilha antes e durante a cahmada da função. Indique os nomes dos registradores e variáveis

armazenados na pilha, marque a localização de SP, e marque claramente cada quadro da pilha

12 Considere a seguinte função em C.

```
// C code
int f(int n, int k) {
  int b;

  b = k + 2;
  if (n == 0) b = 10;
  else b = b + (n * n) + f(n - 1, k + 1);
  return b * k;
}
```

a) Traduza a função f em linguagem Assembly ARM. Preste atenção para salvar e restaurar corretamente os registradores por meio de chamadas de funções e usando registradores convencionados ARM preservados. Comente devidamente seu código. Você pode usar a instrução MUL do ARM. A função deve iniciar no endereço 0x00008100. Mantenha variável local b em R4.

b) Execute a função da parte (a) à mão para o caso de f (2, 4). Desenhe uma imagem da pilha e suponha que SP seja igual a 0xBFF00100 quando f é chamado. Escreva o nome do registro e o valor de dados armazenados em cada local na pilha e acompanhe o valor do ponteiro da pilha (SP). Marque claramente cada quadro de pilha. Você também pode achar útil acompanhar os valores em R0, R1 e R4 durante a execução. Suponha que quando f for chamado, R4 = 0xABCD e LR = 0x00008010. Qual é o valor final de R0?

13) Escreve um trecho de código em ARM assembly que implemente uma progressão aritmética de razão 2, a partir de um valor inicial de entrada e até um valor final (limite superior). Armazene os valores ciclicamente em R4, R5 e R6, de forma que sempre os três ultimos valores da P.A. fique disponíveis.

14) Considere o seguinte trecho de código *assembly* ARM. Os números a esquerda de cada instrução indicam o endereço da instrução.

```
0x000A0028 FUNC1      MOV R4, R1
0x000A002C            ADD R5, R3, R5, LSR #2
0x000A0030            SUB R4, R0, R3, ROR R4
0x000A0034            BL  FUNC2
...                   ...
0x000A0038 FUNC2      LDR R2, [R0, #4]
0x000A003C            STR R2, [R1, -R2]

0x000A0028 FUNC1      MOV R4, R1
0x000A002C            ADD R5, R3, R5, LSR #2
0x000A0030            SUB R4, R0, R3, ROR R4
0x000A0034            BL  FUNC2
...                   ...
0x000A0038 FUNC2      LDR R2, [R0, #4]
0x000A003C            STR R2, [R1, -R2]
```

Liste o modo de endereçamento utilizado em cada linha de código.

15) Faça um programa modularizado que lê uma palavra da memória e identifique a lógica de execução. Caso o byte mais significativo seja:

0xFF – Progressão Aritmética
0xFE – Progressão Geométrica
0xFD – Fibonacci

O segundo byte da palavra indica o primeiro termo da sequência.

O terceiro byte da palavra indica a razão (0x00 em caso de Fibonacci)

O quarto byte indica o termo final.

A sequencia de termos deve ser escrita a partir de um endereço, definido como constante do programa, onde os 3 últimos valores devem corresponder aos 3 últimos dígitos de sua matrícula escolar.

16) Como é organizada a memória em um computador? Mostre a organização no ARM.

17) Como é realizada a conversão de código de um programa de alto nível em linguagem de máquina sem a utilização de uma IDE? Caso o programa seja implementado em uma interface visual, como diagrama ladder ou em blocos, o que mudaria no processo?

18) É possível executar um programa compilado para ARM em um computador x86? Justifique. O que é necessário para permitir a execução?

19) Com relação às instruções especiais na arquitetura ARM, responda
   a) O que são conjunto de instruções Thumb e sua finalidade?
   b) O que são e para que servem as instruções DSP?
   c) Por que existem instruções específicas para operações em ponto flutuante? Cite 2.
   d) Para que servem as instruções WFI, WFE e SEV?
   e) O que é virtualização e qual o papel do Hypervisor?
   f) Quais as principais versões da arquitetura ARM e suas principais características?