MC404: Organização de Computadores e Linguagem de Montagem

Lista de Exercícios

1º semestre de 2013 - Turmas A/B e E

Prof. Edson Borin

Questão 1. É possível saber o valor da flag `V` sabendo o valor de `C` e `Z`? Por que?

Questão 2. Para que utilizamos a instrução `CMP`? Qual sua relação com `SUB`?

Questão 3. Onde a instrução `BL` guarda o endereço da próxima instrução a ser executada quando voltar da função chamada?

Questão 4. Qual a diferença entre as 3 seguintes instruções?

```
MOV PC, LR
MOV R15, LR
MOV PC, R14
```

Questão 5. Qual a diferença entre as formas de armazenamento de dados *little-endian* e *big-endian*? Qual é a forma correta de armazenar dados?

Questão 6. Supondo que o valor em R2 é `3` e o valor em R1 é `10` (ambos em decimal), quais serão os valores desses registradores após esta instrução:

```
ADD R1, R2, R1, LSL #2
```

Questão 7. Qual a diferença entre deslocamento lógico e aritmético? Em uma arquitetura de 4 *bits*, qual o resultado do deslocamento lógico e aritmético de dois *bits* para a direita do número `1010` (binário)? Expresse os resultados em decimal e binário.

Questão 8. Por que é inválido o número `0x101` no campo de imediato em instruções como `ADD R1, R2, 0x101`?

Questão 9. Como poderíamos converter o código abaixo em linguagem de montagem do ARM?

```
if ((x > 3) || (y < 2)){
  x = y;
}
```

Questão 10. Qual a diferença entre `BLT` e `BLO`?

Questão 11. Porque é interessante ler ou escrever dados com tamanhos menores que uma palavra?

a) Economiza espaço em memória.

b) Economiza tempo de processamento.

c) Economiza quantidade de registradores.

d) Todas as anteriores.

e) Nenhuma das anteriores.

Questão 12. O que a seguinte instrução faz?

```
LDRH, R0, [R2], #2
```

Questão 13. O que é ABI?

Questão 14. O que é o modo de endereçamento pré-indexado com *writeback*? Como ele é indicado em uma instrução?

Questão 15. O que são pseudo-instruções? Por que elas existem?

Questão 16. Como dizemos ao montador onde colocar um dado?

Questão 17. Diferencie *Arquitetura* de *Implementação*.

Questão 18. O que são *flags* de *overflow*?

Questão 19. Qual(s) a(s) diferença(s) e semelhança(s) entre as codificações de texto ASCII, ISO-8859-1 (latin-1) e UTF-8?

Questão 20. Por que precisamos de uma instrução `RSC` para subtração e não temos uma equivalente para soma?

Questão 21. Nas arquiteturas ARM, o *bit* 31 de um registrador é o mais ou o menos significativo (MSB ou LSB)?

Questão 22. Para quais tipos de dados da arquitetura ARM (*byte*, *halfword* e *word*) são mapeados os tipos `char`, `short`, `int`, `long` e `long long` da linguagem C?

Questão 23. O que é um barramento? Por que é interessante conectar a memória e dispositivos de processamento gráfico em um barramento exclusivo, distinto dos outros barramentos utilizados para conectar dispositivos como portas serial e paralela e interfaces de rede?

Questão 24. O que é um SoC (*System-on-Chip*)? Quais as vantagens e desvantagens dessa abordagem em relação à implementação de sistemas computacionais com múltiplos circuitos integrados?

Questão 25. O que acontece com o *bit* de *carry* após uma instrução aritmética sem o sufixo "S"?

Questão 26. Como os processadores ARM utilizam o *bit* de *carry* nas instruções ADCS e SBCS?

Questão 27. O *bit* de *carry* é uma das *flags* do processador ARM.

a) Por que não é possível, na linha 3, recuperar o *bit* de *carry* gerado pela linha 1 no código abaixo?

b) Em qual registrador essa *flag* fica armazenada?

c) Quais são as *flags* presentes no processador ARM?

```
1.    ADDS r1, r2, r3
2.    ADDS r4, r5, r6
3.    BEQ rotulo
```

Questão 28. Quantos modos possuem os processadores ARM? Liste-os e descreva as suas funções.

Questão 29. O que significa o termo "salvar o contexto"?

Questão 30. Por que não devemos escrever ou ler dados de um endereço de memória mapeado para dispositivos de E/S quando estamos executando código em modo usuário? Descreva uma situação em que o não cumprimento dessa regra poderia gerar problemas.

Questão 31. Qual instrução do ARM gera uma *trap*?

Questão 32. O trecho de código abaixo desabilita interrupções no processador ARM. Se executado por um programa em modo usuário, ele pode fazer com que o sistema operacional nunca mais retome o controle sobre o processador, pois o dispositivo de relógio (exemplo: GPT) não consegue interromper o processador. Que mecanismos no processador ARM impedem que isso aconteça?

```
msr r0, CPSR
orr r0, r0, #0xC0
msr CPSR, r0
```

Questão 33. Cite três causas para a geração de exceções.

Questão 34. Diferencie interrupções, exceções e *traps*.

Questão 35. O que significa o termo *busy waiting*?

Questão 36. Converta a seguinte sequência de *bytes* representa uma cadeia de caracteres terminada em zero. Quais são as palavras representadas nesta cadeia?

```
4D 43 34 30 34 20 41 42 45 00
```

Questão 37. Qual o problema do código abaixo?

```
LDR r1, =count
LDR r1, [r1] @carrego um contador
comeco:
  BL   foo
  SUB r1, #1
  CMP r1, #0
  BEQ exit
  B comeco
```

Questão 38. Por que precisamos usar sufixos distintos para testar a condição "menor que" quando estamos lidando com números com e sem sinal? Quais são esses sufixos?

Questão 39. O que quer dizer que uma arquitetura usa *memory-mapped IO*?

Questão 40. Escreva um trecho de programa na linguagem de montagem do ARM que realize a soma de dois números de 128 *bits* armazenados nas posições de memória apontadas pelos registradores r1 e r2. O resultado deve ser armazenado na posição de memória apontada por r0.

Questão 41. Quais os registradores salvos automaticamente pelo *hardware* quando ocorre uma interrupção?

Questão 42. Para que serve o *tZIC*?

Questão 43. Ao tentarmos executar uma instrução com *opcode* inválido isso causa uma interrução ou uma exceção?

Questão 44. Enumere os três modos de endereçamento de memória das instruções de *load* e *store* do ARM. Cite as diferenças e descreva suas sintaxes.

Questão 45. Instruções de acesso à memória (leitura e escrita) nos processadores ARM podem vir acompanhadas dos sufixos "B" e "SB". Qual(s) a(s) diferença(s) entre eles?

Questão 46. Escreva código em linguagem de montagem ARM para os seguintes trechos de código na linguagem de alto nível C.

a)
```
for(i=0;i<10;i++)
  y = y + i;
```

b)
```
for(k=10;k>0;--k)
  y = y - k;
```

Questão 47. Estime quanto tempo a aplicação abaixo leva para gerar *overflow* no registrador r1 quando executada em um computador com registradores de 64 *bits* e capacidade para executar 4 bilhões de instruções por segundo.

```
signed long long i;
i = 20;
do{
    y = y + 2;
    i = i + 1;
} while (i>0);
```

```
    MOV r1, 20
while:
    ADD r2, r2, #2
    ADD r1, r1, #1
    CMP r1, #0
    BGE while
```

Questão 48. Qual a diferença entre as instruções BLO e BL?

Questão 49. Qual a vantagem do *UTF-8* com relação ao *ISO-LATIN* e o *MacOSRoman*? E a desvantagem?

Questão 50. Escreva código em linguagem de montagem ARM para o seguinte trecho de código na linguagem de alto nível C. Utilize os modos de endereçamento de memória pré- e/ou pós-indexados.

```
int y[10];
for(i=0;i<10;i++)
  y[i] = i;
```

Questão 51. O que é o `FP`? Qual a sua utilidade? Onde ele é armazenado na arquitetura ARM?

Questão 52. Quantos *bytes* tem cada uma das estruturas abaixo? Justifique.

```
struct s1{
  int x;
  int y;
  char z[256];
}
```

```
struct s2{
  int x;
  int y;
  char *z;
}
```

```
struct s3{
  int x;
  double y;
  float z;
}
```

Questão 53. Segundo a ABI do ARM, a pilha de execução é decrescente e cheia. Descreva uma pilha crescente e vazia. Apresente trechos de código para empilhamento e desempilhamento de elementos.

Questão 54. De quem é o papel de desempilhar os valores armazenados na pilha para passagem de parâmetros? Da função chamada ou da função que a chama (chamadora)?

Questão 55. Nos processadores da família x86 da Intel, não existe um registrador dedicado ao armazenamento do endereço de retorno de uma chamada de função (como o registrador `LR` no ARM). Nessa família, o endereço de retorno é automaticamente salvo na pilha pela instrução para chamada de funções, `CALL`. Cite um caso em que esse comportamento pode gerar perda de desempenho na execução de programas.

Questão 56. Qual a diferença entre variáveis locais e globais, em termos de linguagem de montagem?

Questão 57. O que são chamadas de sistema e por que elas são importantes? Como realizamos chamadas de sistema de acordo com a ABI do ARM? Dê um exemplo.

Questão 58. Como podemos desempilhar 5 palavras armazenadas na pilha com uma única instrução, levando em consideração que não precisamos mais dos dados armazenados nas palavras?

Questão 59. Como você traduziria para linguagem de montagem do ARM o código e estrutura de dados em C:

```
y.valor = &x;
```

Onde `y` e y são variáveis do tipo `struct no`, definida abaixo.

```
struct no{
  int valor;
  struct no * proximo;
}
```

Questão 60. Visando reduzir o espaço ocupado na memória, como poderíamos reescrever este trecho de código?

```
     CMP R1, #10
     BLT else
     ADD R2, R2, #1
     B   fim_se
   else:
     MOV R2, R1
   fim_se:
```

Questão 61. Considere a estrutura em linguagem de montagem do ARM:
`id:  .space 12`
que representa *em C*:

```
struct id{
  int cpf;
  char cod;
  int idade;
}
```

Escreva um trecho de código em linguagem de montagem do ARM para colocar o valor 21 no campo idade.

Questão 62. O que é o vetor de interrupções? Qual a diferença entre armazenar um endereço ou uma instrução no vetor de interrupções? Como funciona o vetor de interrupções dos processadores ARM?

Questão 63. Em qual destes trechos de código estamos passando um valor por referência?

```
Trecho1:
  LDR R0, =a
  BL funcao

Trecho2:
  LDR R0, =a
  LDR R0, [R0]
  BL funcao
```

Questão 64. Quando não é necessário guardar o valor de LR na pilha em uma chamada de função?

Questão 65. Com a seguinte operação: `STMDB SP!, {R4, R5, LR}` qual a ordem dos valores em memória na pilha?

Questão 66. Como podemos determinar qual dispositivo gerou uma interrupção nas placas iMX53 disponíveis no IC-3 e reproduzidas pelo simulador?

Questão 67. Qual o endereço da primeira instrução executada pelo processador ARM após o evento de *reset*? Em um computador com processador ARM, para onde é tipicamente mapeada essa posição de memória?

Questão 68. O que é uma interrupção?

Questão 69. Desviar o fluxo de execução para uma rotina de tratamento, quando uma interrupção ocorre, é papel do *software* ou do *hardware*? E salvar o contexto?

Questão 70. Como alocamos uma variável local na pilha?