MC404 - Organização de Computadores e Linguagem de Montagem
IC - UNICAMP

1º Semestre de 2011 - Turmas A e B

Professor Edson Borin

2ª Prova - Duração: 1:50 h

Nome: _____ RA: _____

Assinatura: _____

1) [30]: O matemático Leonardo Pisa, conhecido como Fibonacci, propôs no século XIII, a seqüência numérica: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …
Essa seqüência, também conhecida como seqüência de Fibonacci, tem uma lei de formação simples: cada elemento, a partir do terceiro, é obtido somando-se os dois anteriores. Veja: 1+1=2, 2+1=3, 3+2=5 e assim por diante.
Escreva a rotina gera_fibonacci, em linguagem de montagem do AVR, para gerar a seqüência de Fibonacci com precisão de 16 *bits*, armazenando-a na memória RAM no formato *little-endian*. A rotina deve parar ao detectar *overflow* da soma de 16 *bits*. A rotina toma como parâmetro o apontador X, que já aponta para o início do vetor que armazenará a seqüência. Lembre-se de comentar o seu código. Utilize o verso desta folha para sua resposta.

2) [10]: Implemente as seguintes macros:
   (a) inicializa_apontador: esta macro toma como parâmetro dois argumentos. O primeiro é o apontador, X, Y ou Z, e o segundo é uma constante de 16 *bits*. A macro deve inicializar o apontador com o valor da constante.

   (b) faca_ate_zero: esta macro toma como parâmetro um label. A macro deve subtrair 1 do conteúdo do inteiro de 16 *bits* armazenado no par de registradores r17:r16 e saltar para o label somente se o resultado da subtração não for zero. A macro deve modificar apenas os registradores r17 e r16.

3) [30]: O programa em linguagem de montagem do AVR abaixo deve computar um vetor de inteiros de 24 *bits* com sinal a partir da soma dos vetores vet_auxiliar e vet_const e entrar em um laço infinito. O vetor vet_const é um vetor de constantes armazenado na memória Flash do AVR. O vetor vet_auxiliar é um vetor computado pela rotina gera_auxiliar e armazenado na memória RAM do AVR. A função gera_auxiliar já foi implementada dentro do arquivo gera_auxiliar.inc. Escreva a rotina soma_vetores para computar a soma dos elementos dos vetores vet_auxiliar e vet_const e armazená-la no vetor vet_resultado. Note que os elementos destes vetores são inteiros de 24 *bits* representados no formato *little endian*. O seu código pode assumir que a constante tamanho_vetor é sempre maior que zero e deve utilizar as macros definidas na questão 2. Se necessário, utilize o verso desta folha para sua resposta.

```
include “m88def.inc”

.EQU tamanho_vetor = 1024  ; Numero de elementos no vetor

.DSEG
  vet_resultado: .BYTE tamanho_vetor*3  ; Vetor resultado
  vet_auxiliar:  .BYTE tamanho_vetor*3  ; Vetor auxiliar

.CSEG
  vet_const: .db 0x12, 0x43, 0x76, ..., 0x23 ; Vetor constante

.ORG 0x0000
  rjmp reset

include “gera_auxiliar.inc”

reset:
  rcall gera_auxiliar
  rcall soma_vetores
  rjmp PC
```

4) [30]: Como vimos em aula, interrupções externas (*External Interrupts*) podem ser disparadas pelos pinos INT0, INT1 ou por qualquer um dos pinos PCINT23...0. O mecanismo de interrupções dos pinos INT0 e INT1 pode ser configurado para disparar interrupções na aresta de subida ou de descida do sinal, dessa forma, é possível, por exemplo, associar um botão de pressão (*push button*) com o pino de entrada INT1 e gerar interrupções toda vez que o botão for pressionado. Escreva um programa completo que conta o número de vezes que o usuário apertou o botão ligado no pino INT1 e o armazene no par de registradores r17:r16 (16 *bits*). Lembre-se de inicializar a pilha e o vetor de interrupções. O seu código pode chamar a rotina inicializa_INT1, no arquivo int1misc.inc, para configurar os registradores EICRA, EIMSK de forma que o mecanismo de interrupções dispare interrupções externas INT1 quando o botão for pressionado.