

## **EA869 – Introdução a Sistemas de Computação Digital**

### **Relatório do Exercício Computacional IV**

**Prof. Levy Boccato – 1º semestre de 2019**

**Bryan Wolff**

**RA: 214095**

**João Pedro Bizzi Velho**

**RA: 218711**

**Julia Alves Farias**

**RA: 200130**

Começamos definindo o endereço inicial onde será escrito os dados na EEPROM “`.equ END_EEPROM = 0x000a`”. Configuramos o vetor de interrupções colocando no primeiro um pulo (`rjmp`) para a posição nomeada como “`config`” (localizado na posição de memória `0x0000`) e na posição relacionada com a interrupção da EEPROM colocamos um pulo para a RSI nomeada `EE_RSI` (localizado na posição de memória `0x002c`). Iniciamos o código a partir da posição `0x0034` que não irá ocupar nenhum vetor, mas onde já será inserido na memória de dados os valores `5, 0x86, 0x73, 0xa4, 0x5b, 0x19`, por exemplo.

Logo após isso, será configurado na posição “`config`” o registrador `ECCR` (usado para controlar a operação da EEPROM), e configuração dos registradores para dados e endereços para memória de programa.

```
config:
sbi ECCR , EEPM1
cbi ECCR , EEPM0
sbi ECCR , EERIE
ldi ZH,high(dados*2)
ldi ZL,low(dados*2)
lpm r1 , Z +
clr r2
ldi r27,high(END_EEPROM)
ldi r26,low(END_EEPROM)
sei
```

Para isso, habilitamos a EEPROM para escrita apenas, configurando os bits `EEPM1` e `EEPM0`, habilitamos também o bit de interrupção da EEPROM (`EERIE`). Em seguida, vamos fazer o registrador `Z` receber a posição de memória de onde será extraído os bytes da memória de programa. O `r1` receberá o número total de bytes a serem escritos, e `r2` receberá o número de bytes que já foram escritos. Já o registrador `X`, recebe o endereço da EEPROM em que gravaremos o primeiro byte. E ao final deste trecho, o comando “`sei`” define a Flag Global Interrupt para permitir interrupções.

O loop “Zilleean” será o responsável irá manter o processador “ocupado” enquanto a EEPROM está sendo gravada, consistindo de uma instrução `NOP` e um `RJMP` para a posição do `NOP`.

Neste ponto todas as configurações para a ocorrência da interrupção da EEPROM já estão completas, portanto o programa ficará no loop Zilleean até que receba o sinal de interrupção vindo da EEPROM.

Quando o sinal da interrupção da EEPROM for emitido a RSI referente à posição `EE_RSI` (`0x0fff`) será

```
EE_RSI:
.org 0x0fff
in r19,SREG
out EEARH , r27
out EEARL , r26
lpm r0 , Z+
out EEDR,r0
inc r26
inc r2
out SREG,r19
sbi ECCR , EEMPE
sbi ECCR , EEPE
cp r1,r2
breq end
```

executada, salvando o registrador de estado SREG em r19 com a instrução `in`, gravamos nos registradores `EEARH` e `EEARL` o endereço da EEPROM no qual gravaremos o byte desejado. Gravamos no registrador `r0` o valor do byte (retirado da memória de programa) e em seguida colocamos `r0` no registrador de dados da EEPROM para ser gravado em seguida incrementamos o registrador (`r26`) que indica a posição da memória de programa de que devemos retirar os bytes, incrementamos o `r2` que indica quantos bytes já foram escritos, restauramos o valor do registrador de estado, setamos o valor dos bits da máscara de interrupção da EEPROM, `EEMPE` e `EEPE`. Por fim, comparamos o valor total de bytes que devem ser gravados e o valor de bytes já gravados, se forem iguais o programa chegou ao fim, o comando “`reti`” retorna ao loop Zillean para continuar a checagem da interrupção

```
retorno:
    reti
end:
    cbi EECR, EERIE
    break
```

Observação: podemos ter duas opções para finalizar o código, sendo elas: Manter o loop de checagem de interrupção (apesar da interrupção referente à interrupção já ter sido desabilitada) indefinidamente ou parar a execução com um `break` dentro da RSI.