CEFET-MG — Centro Federal de Educação Tecnológica de Minas Gerais

DEPARTAMENTO DE COMPUTAÇÃO DE DIVINÓPOLIS — DECOM-DV

Microprocessadores e Microcontroladores

Primeira Atividade Avaliativa

Aluno: BÁRBARA BRAGA GUALBERTO CORREA

Valor: 30 pontos (cada questão vale 5 pontos)

Turma: 2024/1

Prof. M. Sc. Diego Ascânio Santos

Respostas:

1	2	3	4	5	6

Questão 1

(ENADE 2005 - 11) Apesar de todo o desenvolvimento, a construção de computadores e processadores continua, basicamente, seguindo a arquitetura clássica de von Neumann. As exceções a essa regra encontram-se em computadores de propósitos específicos e nos desenvolvidos em centros de pesquisa. Assinale a opção em que estão corretamente apresentadas características da operação básica de um processador clássico:

a) Instruções e dados estão em uma memória física única; um programa é constituído de uma seqüência de instruções de máquina; uma instrução é lida da memória de acordo com a ordem dessa seqüência e, quando é executada, passa-se, então, para a próxima instrução na seqüência.

b) Instruções e dados estão em memórias físicas distintas; um programa é constituído de um conjunto de instruções de máquina; uma instrução é lida da memória quando o seu operando-destino necessita ser recalculado; essa instrução é executada e o resultado é escrito no operando de destino, passando-se, então, para o próximo operando a ser recalculado.

c) Instruções e dados estão em uma memória física única; um programa é constituído de um conjunto de instruções de máquina; uma instrução é lida da memória quando todos os seus operandos-fonte estiverem prontos e disponíveis; essa instrução é executada e o resultado é escrito no operando de destino, passando-se, então, para a instrução seguinte que tiver todos seus operandos disponíveis.

d) Instruções e dados estão em memórias físicas distintas; um programa é constituído de um conjunto de instruções de máquina; uma instrução é lida da memória quando todos os seus operandos-fonte estiverem prontos e disponíveis; essa instrução é executada e o resultado é escrito no operando de destino, passando-se, então, para a instrução seguinte que estiver com todos os seus operandos disponíveis.

e) Instruções e dados estão em memórias físicas distintas; um programa é constituído de uma seqüência de instruções de máquina; uma instrução é lida da memória de acordo com a ordem dessa seqüência e, quando é executada, passa-se, então, para a próxima instrução na seqüência.

Questão 2

Raul deseja fazer um LED comutar de estado a cada 5 segundos. Sem muito critério do entendimento viu nos códigos dos slides do prof. Ascânio o seguinte ISR que fazia o LED comutar de estado a cada 5 segundos:

Rotina de interrupção de Overflow associada ao `Timer2`

```
ISR(TIMER2_OVF_vect) {
```

```
    overflows ++;
    if (overflows == 306) {
        overflows = 0;
        // Toggle the LED state
        digitalWrite(13, !digitalRead(13));
    }
}
```

Entretanto, ao copiar os códigos, não se atentou ao modo correto de funcionamento e configurou — sem querer — o prescaler do `Timer2` para o fator de 256, através do registrador de controle `TCCR2B` pela seguinte instrução na função `setup()`:

```
    TCCR2B = 0b00000110;
```

Com isso, em vez do LED comutar de estado a cada 5 segundos, ele comutava a cada 1.25 segundos (aproximadamente). Qual dos seguintes modos de *prescaling* do `Timer2` faz com que o LED comute de estado a cada 5 segundos, como deseja Raul?

a) `TCCR2B = 0b00000000;`

b) `TCCR2B = 0b00000001;`

c) `TCCR2B = 0b00000011;`

d) `TCCR2B = 0b00000111;`

e) `TCCR2B = 0b00000110;`

Questão 3

Cláudia perguntou ao professor Ascânio se sua rotina de interrupções de overflow do `Timer2` estava correta para contar intervalos de tempo de 10 em 10 segundos, pois, estava com duvidas se seus cálculos de quantidade de *overflows* e o modo de *prescaling* que havia definido para o `Timer2` estavam corretos. O professor Ascânio verificou o código apresentado por Cláudia e disse que tanto os cálculos quanto o *prescaling* estavam corretos, mas, que a rotina de interrupção de *overflow* do `Timer2` — *Flag* `TOIE2` — estava desabilitada.

Qual deve ser a instrução que Cláudia deve adicionar à função `setup()` de seu código para habilitar a interrupção de *overflow* do `Timer2`?

a) `TIMSK2 = TIMSK2 | 0b00000000;`

b) `TIMSK2 = 0b00000000;`

c) `TIMSK2 = TIMSK2 | 0b00000001;`

d) `TIMSK2 = TIMSK2 | 0b00000010;`

e) `TIMSK2 = TIMSK2 | 0b00000100;`

Questão 4

Considerando a seguinte rotina de interrupção de *overflow* associada ao `Timer2`, que o `Timer2` está configurado para operar no modo normal (modo timer) e que a frequência do *clock* do `Timer2` é de 16 MHz, qual fator de prescaler faz com que o estado do LED conectado ao pino digital 13 comute a (aproximadamente) cada 5 segundos?

Rotina de interrupção de Overflow associada ao `Timer2`

```
ISR(TIMER2_OVF_vect) {
    overflows ++;
    if (overflows == 306) {
        overflows = 0;
        // Toggle the LED state
        digitalWrite(13, !digitalRead(13));
    }
}
```

a) 1.

b) 16.

c) 64.

d) 256.

e) 1024.

Questão 5

Quanto a entradas digitais no Arduino, é correto afirmar que:

a) Esperar um tempo após a leitura de um pino digital — técnica conhecida como **debounce** — é uma boa prática para garantir que o valor lido seja estável.

b) A comutação de chaves mecânicas é imune ao aparecimento de ruídos, efeito conhecido como **bouncing**.

c) Resistores de *pull-up* fazem com que o estado padrão de uma entrada digital seja nível lógico BAIXO.

d) Resistores de *pull-down* fazem com que o estado padrão de uma entrada digital seja nível lógico ALTO.

e) Nenhum dos itens anteriores está correto.

Questão 6

A respeito do ambiente de desenvolvimento do Arduino, da sua linguagem de programação *sketch* e de demais conceitos relacionados, julgue os itens a seguir.

I. A função `setup()` é executada uma única vez, quando o programa é inicializado;

II. A função `setup()` é usada para inicializar configurações e preparar o estado inicial do programa;

III. A função `loop()` é executada continuamente, em um loop infinito, até que o microcontrolador seja desligado;

IV. A instrução `pinMode(3, INPUT_PULLUP);` configura o pino 3 como entrada digital com nível lógico invertido;

V. O Arduino pode acionar diretamente atuadores — elementos de carga — de baixa potência e que suportem o nível de tensão de saída do microcontrolador;

VI. Para atuadores de média / alta potência é necessário o uso de circuitos auxiliares de acionamento comumente baseados em transistores.

Estão incorretos:

a) Nenhum item está incorreto.

b) I, III e V.

c) II, IV e VI.

d) I, II, III, V e VI.

e) Todos os itens estão incorretos.