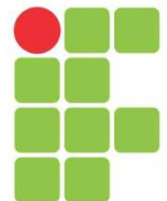


Sistemas Embarcados - Introdução

Instituto Federal de Pernambuco
Coordenação de Informática/IFPE

Anderson L. S. Moreira
`anderson.moreira@recife.ifpe.edu.br`





Attribution – ShareAlike 3.0

Você é livre para

- Copiar, distribuir, mostrar, e adaptar o trabalho
- Para fazer trabalhos derivados
- Para fazer uso comercial do trabalho

Seguindo certas condições

Atribuição. Você deve dar os devidos créditos ao autor original.

Compartilhar. Se você altera, transformar ou construir em cima deste trabalho, você deverá distribuir o trabalho resultante somente sobre uma licença idêntica a está.

Para qualquer reuso ou distribuição, você deve deixar claro aos outros os termos de licença deste trabalho.

Qualquer destas condições podem ser modificadas se você tiver permissão do autor original.

Se uso e outros direitos não são afetados pelas regras acima.

License text: <http://creativecommons.org/licenses/by-sa/3.0/legalcode>

- Berkeley Design technology, Inc.: <http://www.bdti.com>
- EE Times Magazine: <http://www.eet.com/>
- Linux Devices: <http://www.linuxdevices.com>
- **Embedded Linux Journal:**
<http://www.linuxjournal.com/tag/embedded>
- Embedded.com: <http://www.embedded.com/>
 - Embedded Systems Programming magazine
- Circuit Cellar: <http://www.circuitcellar.com/>
- Electronic Design Magazine: <http://electronicdesign.com/>
- Electronic Engineering Magazine:
<http://www.computeroemonline.com/>
- Sensors Magazine: <http://www.sensorsmag.com>
- Embedded Systems Tutorial: <http://www.learn-c.com/>
- Collections of embedded systems resources
 - <http://www.ece.utexas.edu/~bevans/courses/ee382c/resources/>
 - [http://www.ece.utexas.edu/~bevans/courses/realtime/resources.h
tml](http://www.ece.utexas.edu/~bevans/courses/realtime/resources.html)
- **Embedded Linux Experts:** <http://free-electrons.com/>
- **Simulador RI Tools disponível em:** <http://va.mu/BPWop>

- Trabalhos de casa: 15%
- Projeto (2-3 alunos por projeto): 20%
- 1º Exame (teórico): 15%
- 2º Exame (apenas prático): 50%



Sistemas Embarcados

INTRODUÇÃO

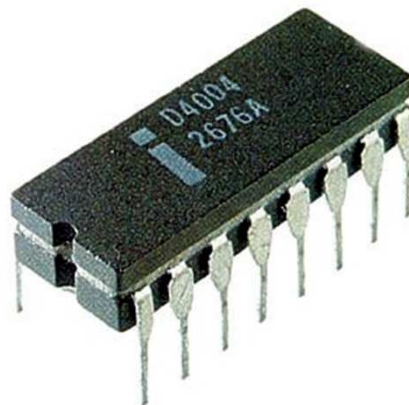
- De acordo com as previsões caracteriza-se por a termos como:
 - Era Pós-PC;*
 - Disappearing computer*
 - Ubiquitous computing*
 - Pervasive computing*
 - Ambientes inteligentes
 - Sistemas embarcados ou Sistemas Embutidos (*Embedded systems*)



- Período de 1940 – 1950
 - Primeiros computadores ocupavam grandes dimensões de espaço;
- Período de 1960 – 1970
 - Começa a ficar pequeno o tamanho dos computadores;
 - Intel 4-bit 4004 (1971) – Marcou o aparecimento do microprocessador (micro significa pequeno)
- Período de 1970 – 1980
 - Surgimento dos PC's;
 - Primeiros dispositivos em outros aparelhos.



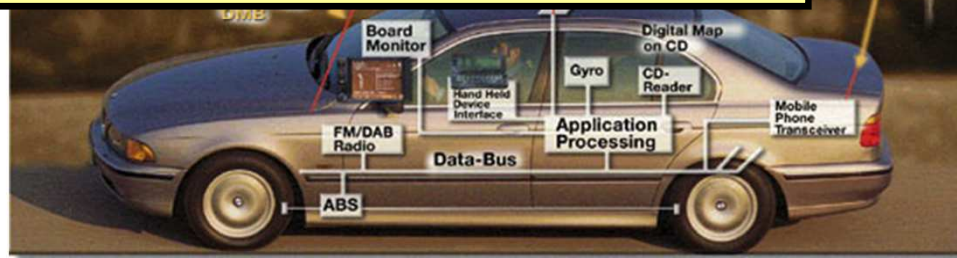
- A diminuição de tamanho permitiu microprocessadores serem incorporado a outros dispositivos elétricos
 - Máquinas de lavar roupa, fornos de micro-ondas e máquinas POS;
- Na verdade, o primeiro microprocessador foi projetado para uma calculadora de impressão.
- Na década de 1990, esses sistemas se tornaram conhecido como Sistemas Embarcados.



- Notável crescimento do uso de sistemas computacionais em diferentes tipos de aplicações



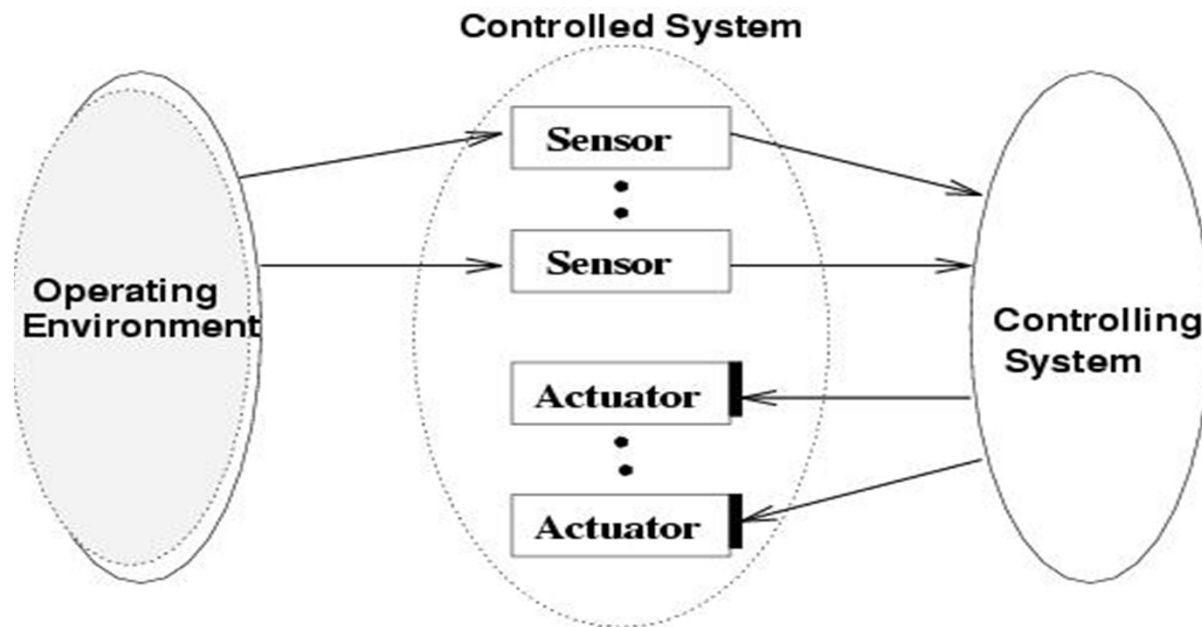
-Sistemas cada vez menores





- Definição clássica:
 - Sistema embarcado é todo aquele sistema que está embarcado em outro sistema. (Vahid, 2002)
- Estes sistemas possuem, no geral, uma ou poucas funções dedicadas.
- São frequentemente criados sob rigoroso processo de construção:
 - potência, desempenho, tamanho, tempo e restrições de custo.

- Geralmente tem de reagir rapidamente às mudanças do ambiente em que está inserido e gerar novas respostas de saída;



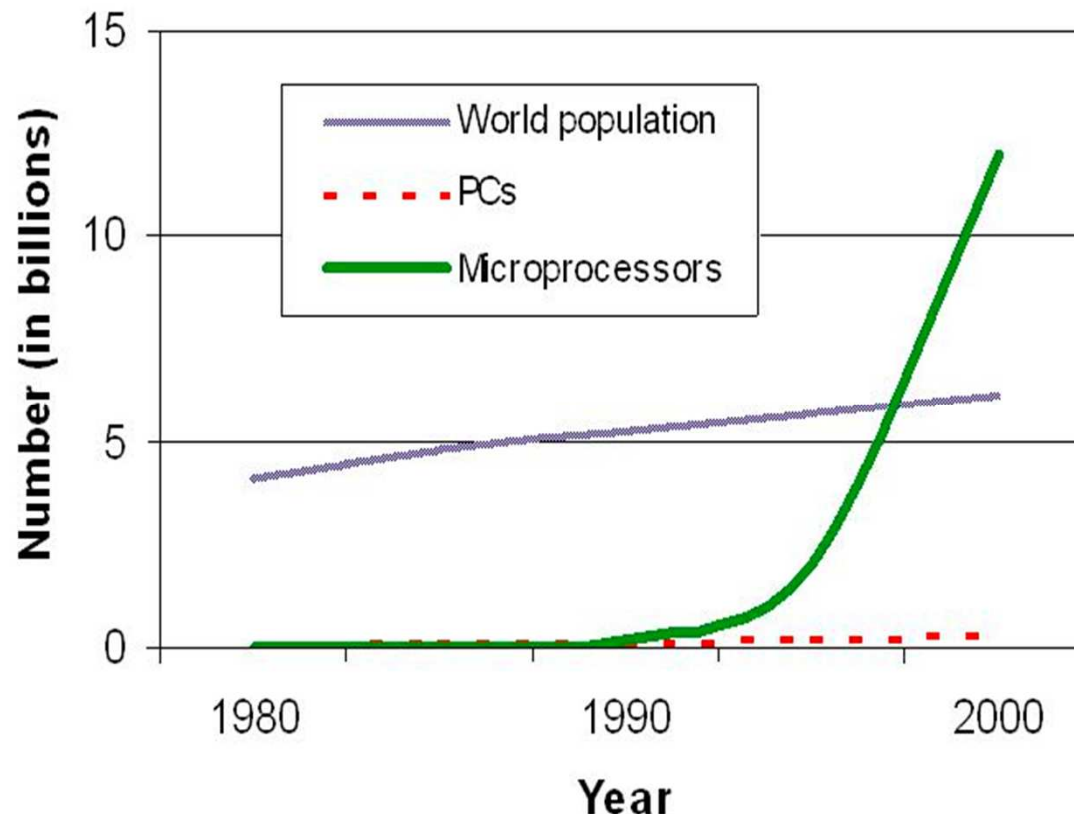
- Além de PCs, laptops e servidores, a maioria dos sistemas que operam com energia elétrica e fazem algo inteligente possuem sistemas embarcados;

- Quais destes são sistemas embarcados?

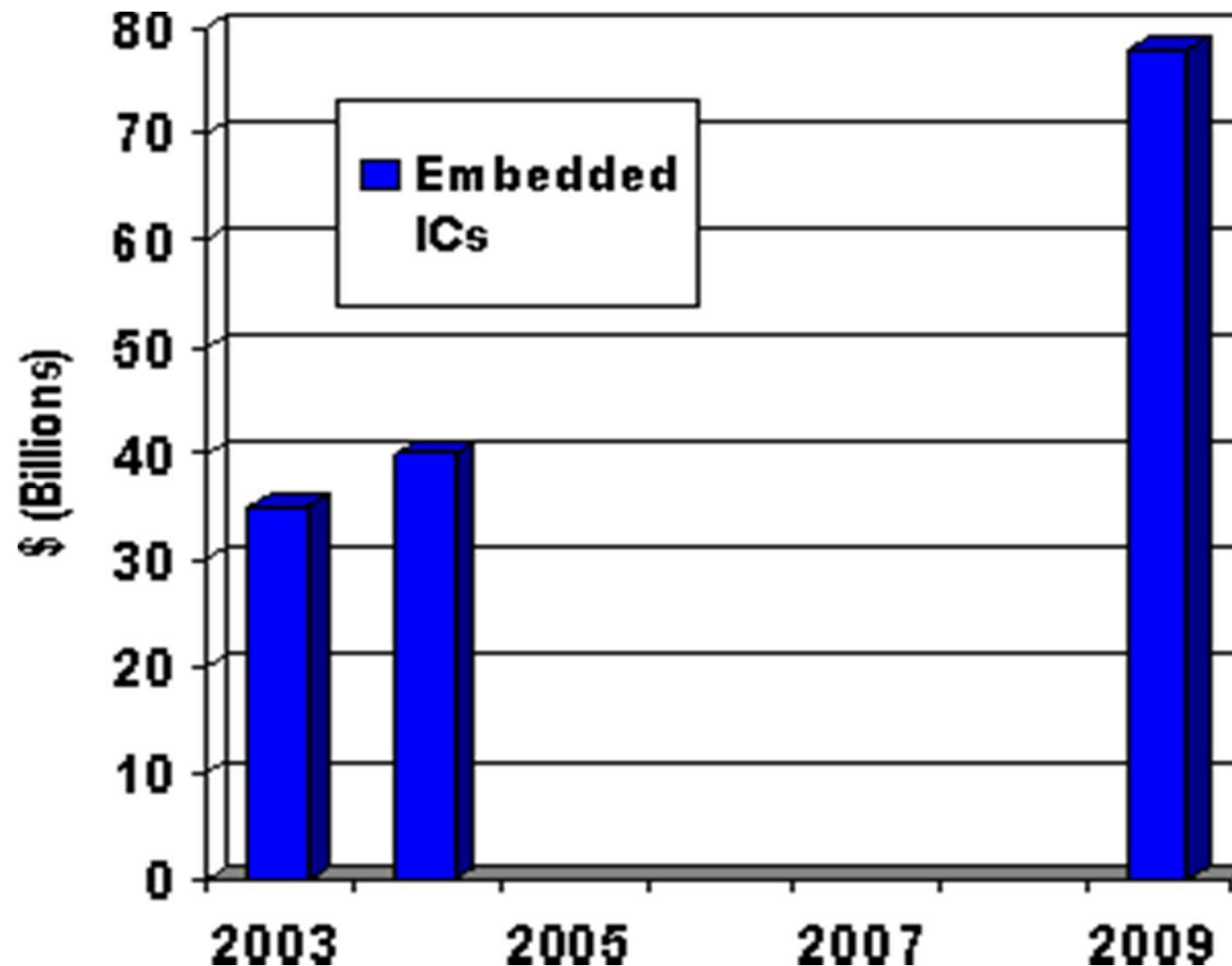
#	Questão	Sua resposta
1	Desktop PC	Verdadeiro
		Falso
2	Bateria eletrônica (instrumento musical)	Verdadeiro
		Falso
3	Elevador	Verdadeiro
		Falso
4	Cloud server	Verdadeiro
		Falso
5	Marca-passos cardíaco	Verdadeiro
		Falso

- Cada ano cerca de 10 bilhões de microprocessadores são fabricados. Destes $\sim 98\%$ são sistemas embarcados (Alan Burns);
- Circuitos integrados (CI ou chips) em que os microprocessadores são implementados, vem dobrando a sua capacidade de transistores a cada 18 meses (Lei de Moore);
- Essa duplicação significa:
 - Que o sistema tem o mesmo tamanho e tem cada vez mais capacidade
 - telefone celular
 - Que o mesmo dispositivo pode ser menor (metade de 18 em 18 meses) permitindo assim novas invenções
 - pílulas computadorizadas que podem ser ingeridas

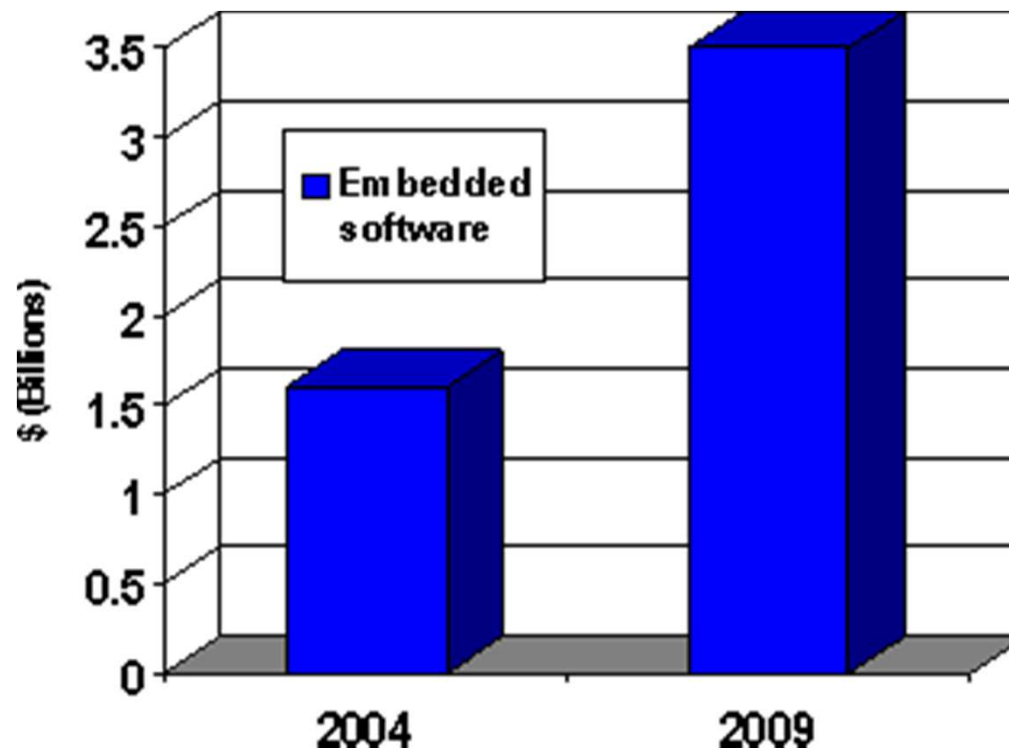
- Os microprocessadores produzidos por ano estão crescendo de forma exponencial, principalmente destinado para sistemas embutidos. ([Study of Worldwide trends and RD programmes in Embedded Systems by FAST GmbH.](#))

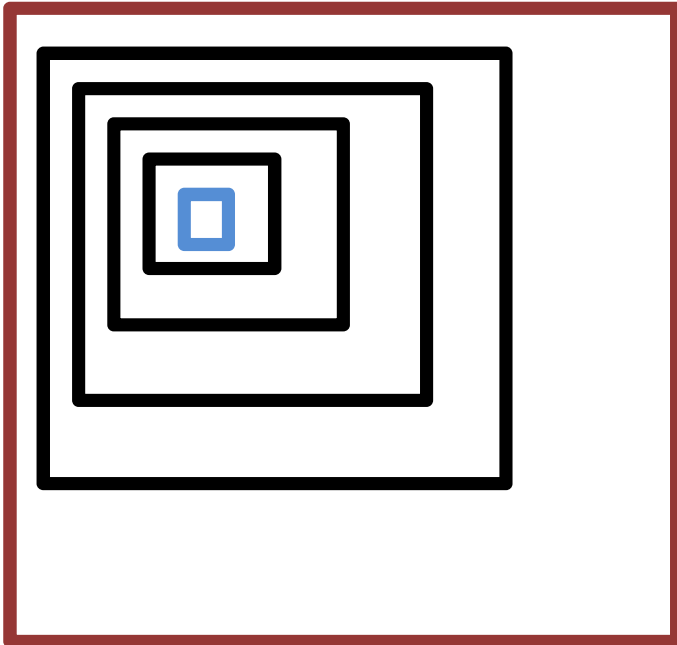


- As vendas brutas do CI's destinados a sistemas embarcados estão crescendo a cada ano. (BBC, Inc.)



- Vendas brutas de Sistemas Embarcados também vem crescendo rapidamente a cada ano (NY Times, 2010)





Cada caixa: 1,5 anos

- 10 M de transistores
Um processador básico de 32 bits

1990:
10 milhões de
transistores em 1 cm
quadrado de chip

- 1999:
10 milhões de
transistores em 0,015 cm
quadrado de chip
1/64 ou 1,5% do original em 9 anos

- 2008:
0,00024 cm quadrado de
chip

2015:
0,0000038 cm quadrado
de chip – Nem é visível
aqui

- Considere um chip num smartphone hoje em dia. Se a lei de Moore prevalece, esse chip em 6 anos iria armazenar quantas vezes mais transistores?
- A capacidade iria dobrar 4 vezes em 6 anos, $2*2*2*2 = 16$. Então em 6 anos este smartphone iria ter 16 vezes mais capacidade!
- Considere um chip num smartphone hoje em dia. Se a lei de Moore prevalece, esse chip em 12 anos iria ocupar que fração do tamanho original? Escreva a resposta como uma fração: Ex. $\frac{1}{4}$.
- $\frac{1}{2}$ em 1,5 anos; $\frac{1}{4}$ em 3 anos; $\frac{1}{8}$ em 4,5 anos e assim vai. Logo em 12 anos seria $\frac{1}{256}$!

- O que são Sistemas Embarcados?
- O que é um Sistema de Tempo Real?
- O que são Sistemas Reativos?
- Você diria que todo S.E. é também um Sistema de Tempo Real?
E é um Sistema Reativo?
- É fundamental a execução de tarefas em paralelo em um S.E.? Porquê?
- Como você compara o mercado de S.E. em relação ao de desktops?

Sistemas Embarcados

COMPONENTES BÁSICOS

- Um sistema com componentes elétricos usa fios com sinais de voltagem contínua;
- Uma abordagem abstrata e útil é considerar:
 - Baixa voltagem (0 volts até 0,3 volts) = 0;
 - Alta voltagem (0,7 volts até 1,2 volts) = 1.
- Um bit (acrônimo de "*binary digit*") é um dígito com dois estados possíveis;
- A mudança do valor de um bit através do tempo é chamada de **sinal digital**;
- Digital refere-se ao sinal ter valores discretos do que valores contínuos.

- Um **switch** (chave) é um componente eletromecânico com um par de contatos elétricos;
- O contato estão em um dos dois estados mecanicamente controlados:
 - **Fechado**
 - **Aberto**
- Quando fechado os contatos estão eletricamente conectados;
- Quando aberto os contatos estão eletricamente desconectados.

- Um projeto de sistema digital ajuda a pensar na abstração de um switch;
- Um switch é um componente com um simples bit de saída;
- Este pode se 0 ou 1, dependendo em que posição se encontra o recurso.



Abstração de um switch

- Um push button opera semelhante a um switch;
- Tem um par de contatos elétricos e dois estados mecânicos controlados;
 - Aberto e Fechado
- Ao contrário do switch essa estrutura entra no estado de fechado ao ser pressionado;
- No momento em que a força de pressão é removida este entra em estado de aberto.



Abstração de um botão

- Um LED (Light Emitting Diode) é um semicondutor com um par de contatos;
- Quando uma pequena corrente passa pelos contatos do LED este ilumina;
- Uma abstração de um LED é um componente com um bit simples como entrada que pode ser 0 ou 1:
 - Quando a entrada é 0 o LED não ilumina;
 - Quando a entrada é 1 o LED ilumina.



Abstração de um LED

- Podemos construir sistemas simples que é composto de um switch e um LED conectados;



- Quando o botão é deslizado o LED irá iluminar.

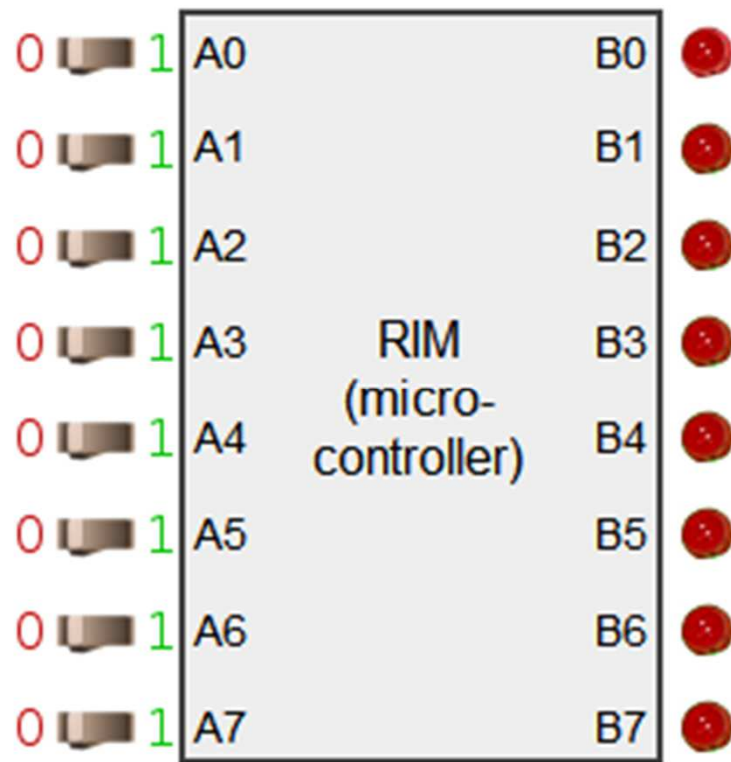
Atenção

- Este sistema está aquém de ser um sistema embarcado porque falta-lhe funções computacionais. Por exemplo, o sistema não pode ser facilmente modificado para acender o LED quando qualquer um dos dois interruptores são definidos para ligar. Um elemento que pode executar algumas funções computacionais é uma parte fundamental de um sistema embarcado.

- Um microcontrolador é um componente programável que lê entradas digitais e escreve saídas digitais de acordo com algum programa interno armazenado que o computa;
- Centenas de diferentes microcontroladores são comercialmente disponíveis:
 - PIC; 8151; 68HC11; AVR; Atmel;
- Um microcontrolador contém uma memória interna programável que armazena um código gerado de operações assemblers em linguagens tipo C, C++, Java ou Assembly.



- O simulador de microcontrolador utilizado na disciplina é o RIM (Riverside-Irvine Microcontroller);
- Consiste:
 - Oito bits de entrada
 - Identificados por A0, A1, ..., A7
 - Oito bits de saída
 - Identificados por B0, B1, ..., B7
- RIM é capaz de executar código em C que podem acessar essas entradas e saídas como variáveis globais.

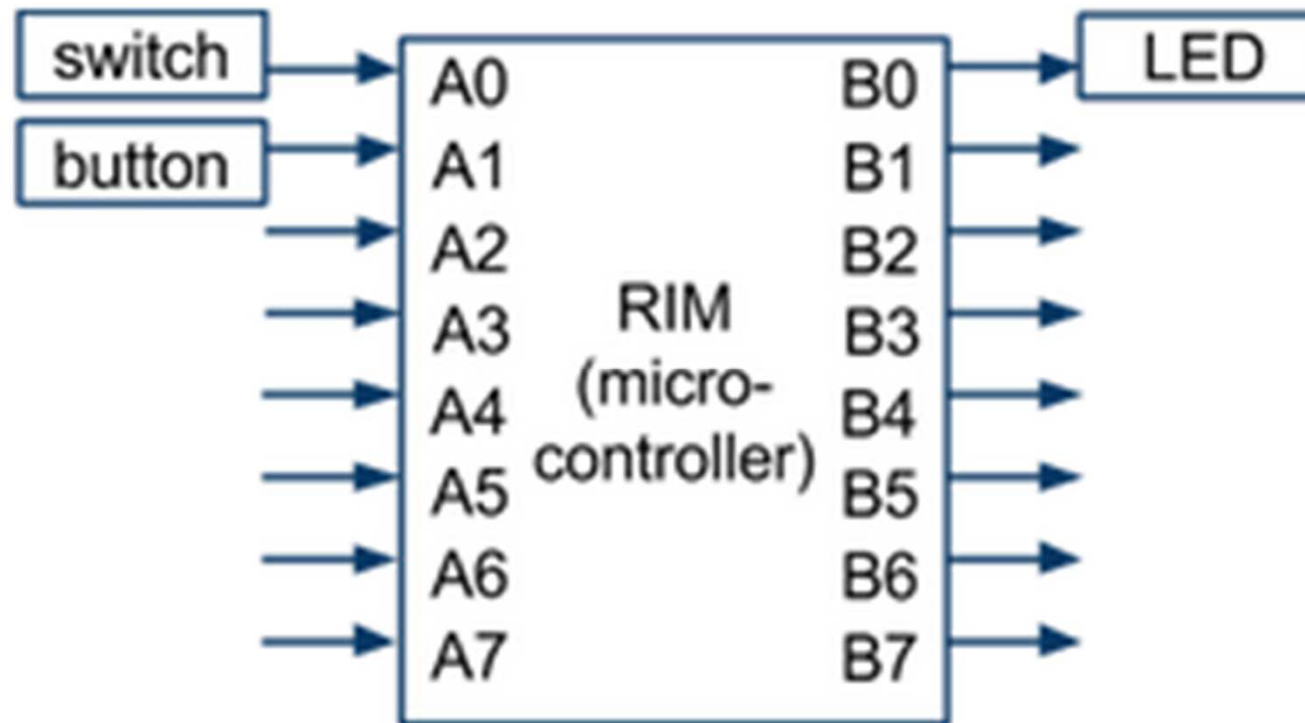


```
#include "RIMS.h"

int main(void) {
    while (1) { // Repeat forever
        B0 = A2 && A1 && A0;
    }

    return 0; // Never reached
}
```

- No exemplo a instrução $B0 = A2 \ \&\& \ A1 \ \&\& \ A0$ seta a saída do microcontrolador B0 para 1 se as entradas A2, A1 e A0 forem todas 1;
- O loop `while (1) { <statements> }` é um recurso comum em C para sistemas embarcados
 - **Loop Infinito** → Repetir continuamente a instrução contida no microcontrolador
- Podemos utilizar um microcontrolador para adicionar funcionalidades e assim criar um sistema embarcado:
 - **Switch** e **push buttons** são exemplos de sensores e não de Sistemas Embarcados. Estes convertem fenômenos físicos em saídas digitais;
 - O **LED** é um exemplo de um atuador, que converte saídas digitais de sistemas embarcados em fenômenos físicos.

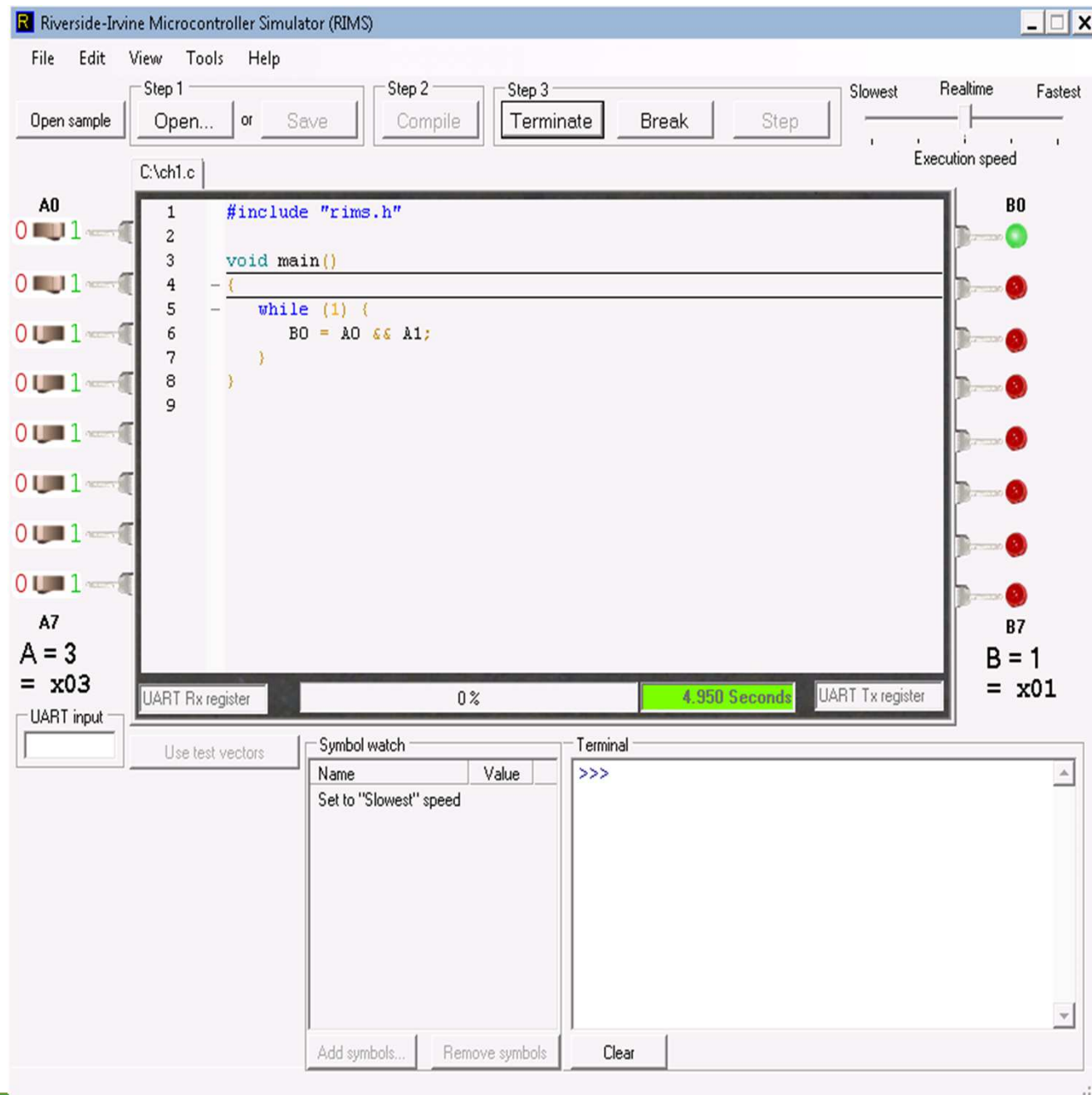


O RIM conectado a switches, buttons e LED's

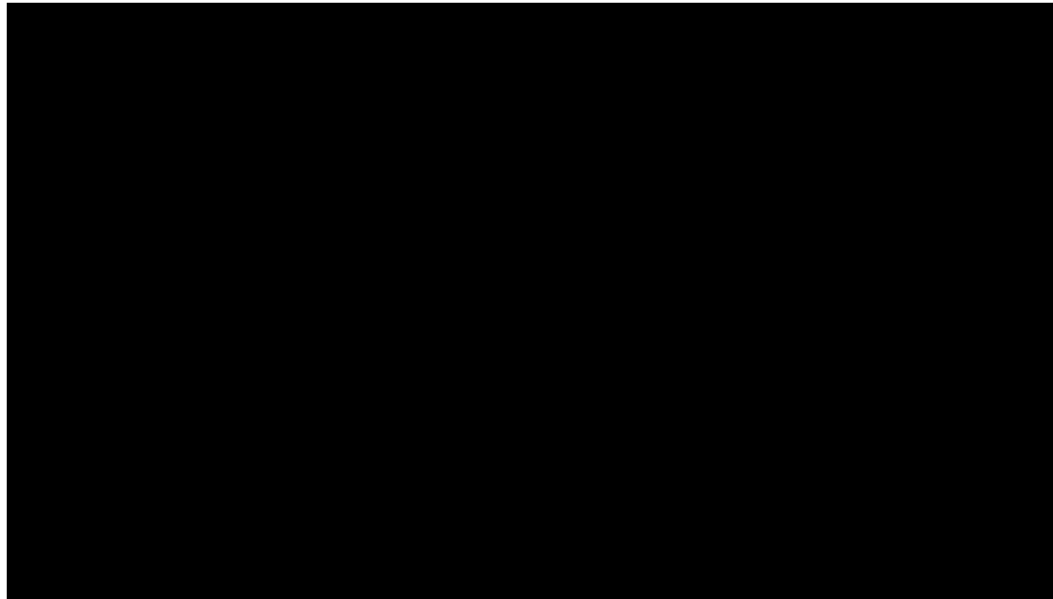
#	Questão	Resposta
1	Um push button tem como saída 1 Volt quando pressionado, 0 Volt quando liberado	Verdadeiro
		Falso
2	Um LED é projetado para ser ligado quando a voltagem de entrada ultrapassar 1 Volt	Verdadeiro
		Falso
3	Um microcontrolador executa operações que converte valores de entrada em valores de saída	Verdadeiro
		Falso
4	Um microcontrolador não pode ser programado	Verdadeiro
		Falso

Sistemas Embarcados

RIM-SIMULATOR



- RIMS (RIM Simulator) é uma ferramenta gráfica baseada em PC que suporta programação em C e simula um microcontrolador de 8 bits;
- As oito entradas **A0–A7** são conectadas por oito switches, em que cada um pode ser inicializado por **0** ou **1**;
- As oito saídas **B0–B7** são conectadas por oito LED's em que tornam-se **vermelho** quando a saída é **0** e **verde** quando for **1**.



- O código em C pode ser escrito no centro, na caixa de texto;
- A linha `#include "rims.h"` é requerida no topo de qualquer arquivo escrito na ferramenta;
- Aperte o botão **SAVE** para salvar o código em C;
- Aperte o botão **COMPILE** para traduzir o código em C para código de execução de máquina;
 - Por padrão é escondido para o usuário
- Quando finalizado aperte **TERMINATE**;
- Pode clicar com o botão direito em um dos componentes **switch** ou **LED** para alterar respectivamente em **button** ou **speaker**.

- Baixe, instale e execute o RIMS;
- Note que o programa em C padrão aparece no centro da ferramenta;

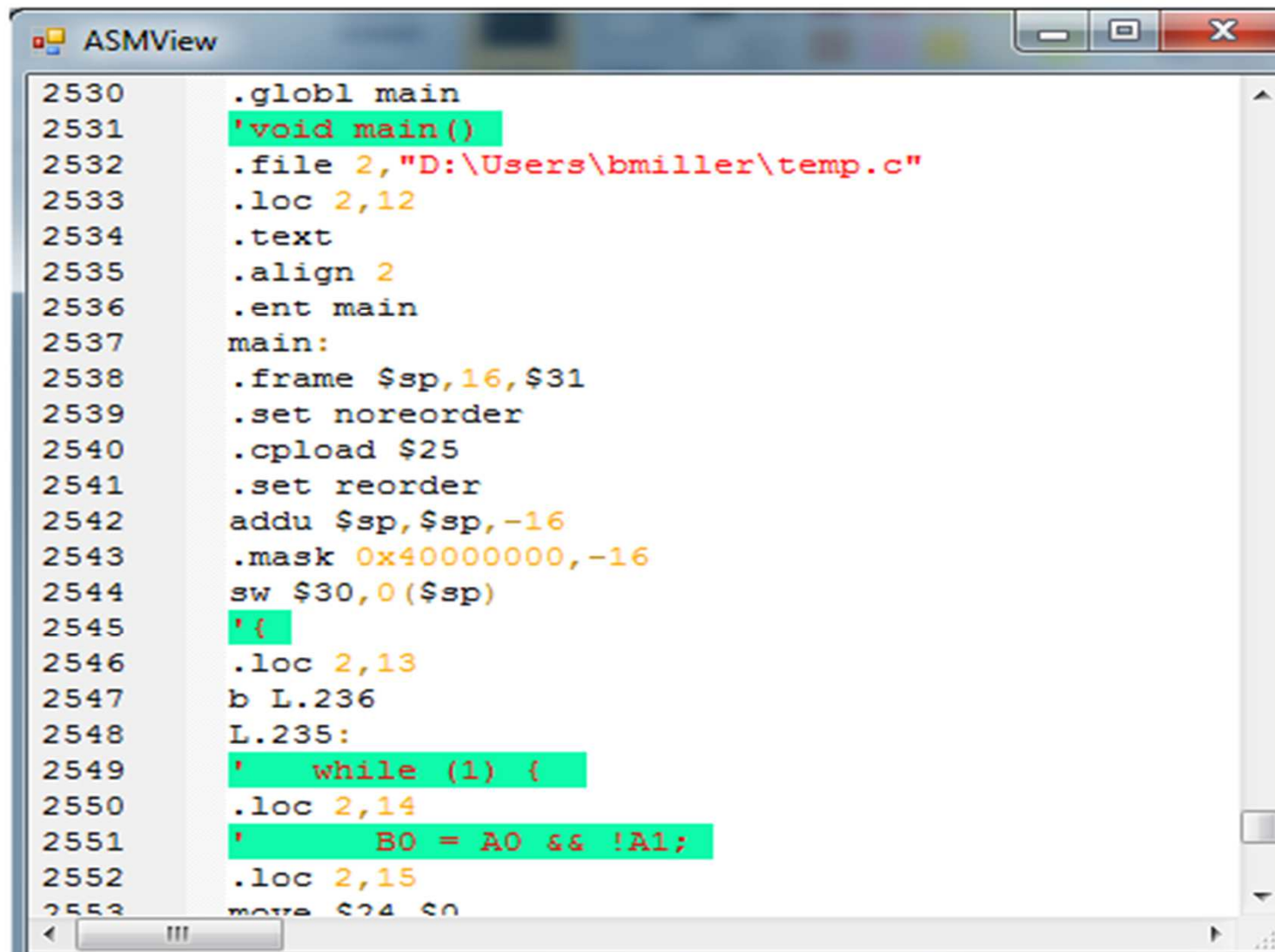
```
#include "RIMS.h"

void main()
{
    while (1) {
        B0 = A1 && A0;
    }
}
```

- Substitua o arquivo para o que deseja executar;
- Lembrando dos comandos básicos:
 - SAVE → COMPILE → RUN

- Escrever um programa em C na ferramenta RIM que define $B0 = 1$ quando o número de 1s no A2, A1, A0 é de dois ou mais (ou seja, quando A2 A1 A0 são 011, 110, 101 ou 111). Dica: Use operador lógico “ou” (||) além de sua lógica pessoal. Executar o programa.

- Apertando o botão BREAK temporariamente para a execução;
- Botão muda para CONTINUE e aparece uma seta para a próxima instrução em C;
- Cada vez que aperta STEP este executa uma instrução de cada vez;
- A caixa embaixo do código mostra quantos segundos o programa em C está em execução.



```
2530 .globl main
2531 'void main()
2532 .file 2,"D:\Users\bmiller\temp.c"
2533 .loc 2,12
2534 .text
2535 .align 2
2536 .ent main
2537 main:
2538 .frame $sp,16,$31
2539 .set noreorder
2540 .cplod $25
2541 .set reorder
2542 addu $sp,$sp,-16
2543 .mask 0x40000000,-16
2544 sw $30,0($sp)
2545 '{
2546 .loc 2,13
2547 b L.236
2548 L.235:
2549 ' while (1) {
2550 .loc 2,14
2551 ' B0 = A0 && !A1;
2552 .loc 2,15
2553 move $24,$0
```

Sistemas Embarcados

DIAGRAMA DE TEMPO

- Um sistema embarcado opera continuamente no tempo. Uma representação comum de como o sistema opera (ou deveria operar) é um diagrama de tempo;
- Diagrama de Tempo mostra o tempo fluindo pela direita e desenha os valores dos sinais de bit como 1 (alto) ou 0 (baixo);
- Vamos considerar o exemplo do $B0 = A1 \ \&\& \ A0$;

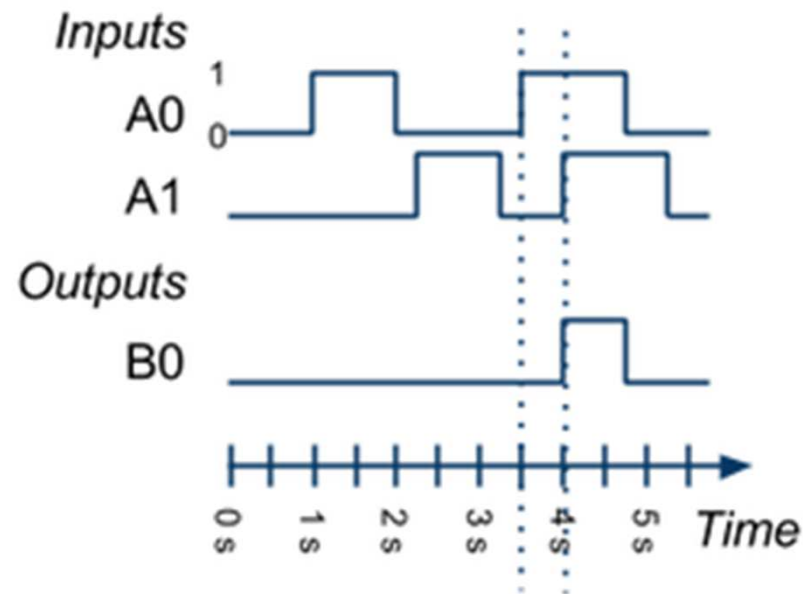


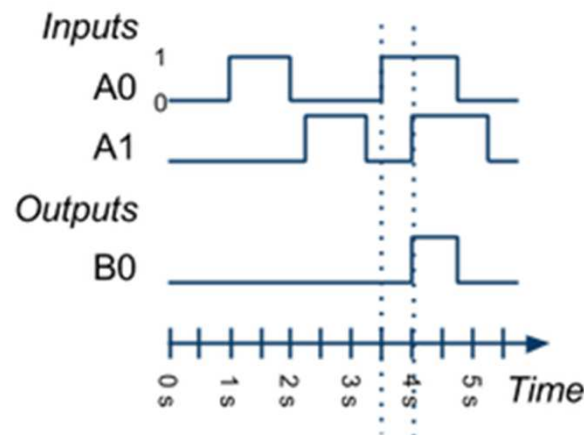
Diagrama de Tempo - Duas entradas e uma saída

- A0 é 0 do tempo 0 ms a 1 ms, quando este muda o valor para 1. A0 continua 1 até 2 ms, quando muda o valor para 0. E assim continuamente;
- Os valores 1 e 0 estão marcados para o sinal A0, mas são usualmente implícitas.

- O diagrama de tempo mostra que B0 é 1 durante o intervalo de tempo quando ambos A0 e A1 são 1, ou seja entre 4 ms e 5 ms;
- As linhas verticais são às vezes adicionadas para ajudar a mostrara quantos itens são marcados ou para criar regiões de tempo distintas;

- Exercício:
 - Desenhe um diagrama de tempo mostrando todas as possíveis combinações de três entradas A0, A1 e A2 do exemplo feito anteriormente (slide 41). Use linhas verticais para indicar as áreas de mudança.
 - Um programa deve se comportar da seguinte maneira: Setar B0 com 1 se exatamente uma das entradas A0 ou A1 é 1. Desenhe o diagrama de tempo ilustrando esse comportamento

- Uma mudança no sinal é chamado de **evento**;
- Evento é tipicamente referenciado a uma mudança de sinal de 0 para 1 ou de 1 para 0.
 - Mas pode referenciar a mudanças de um inteiro (multi bit) ou de outro sinal.

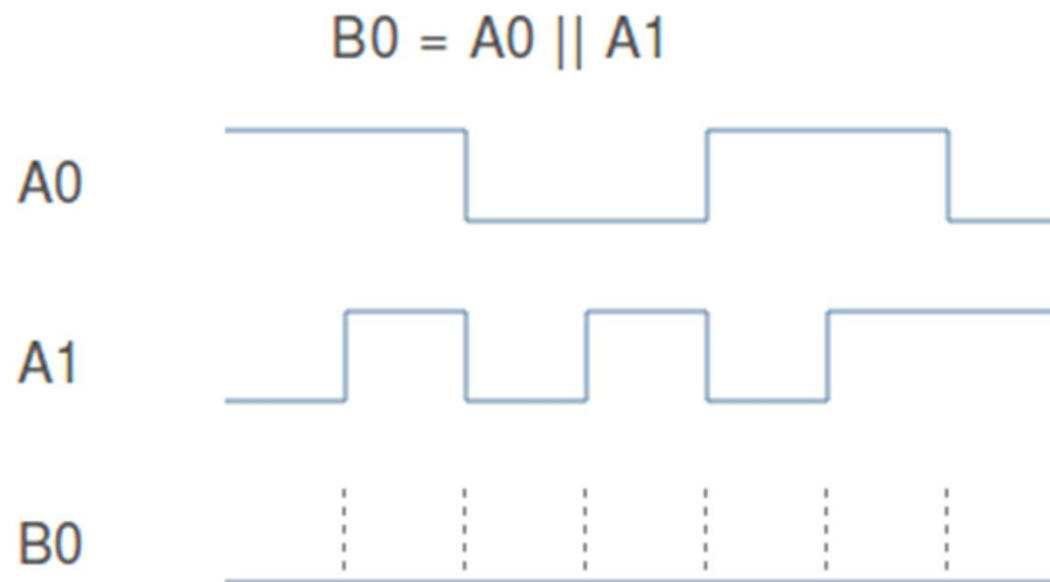


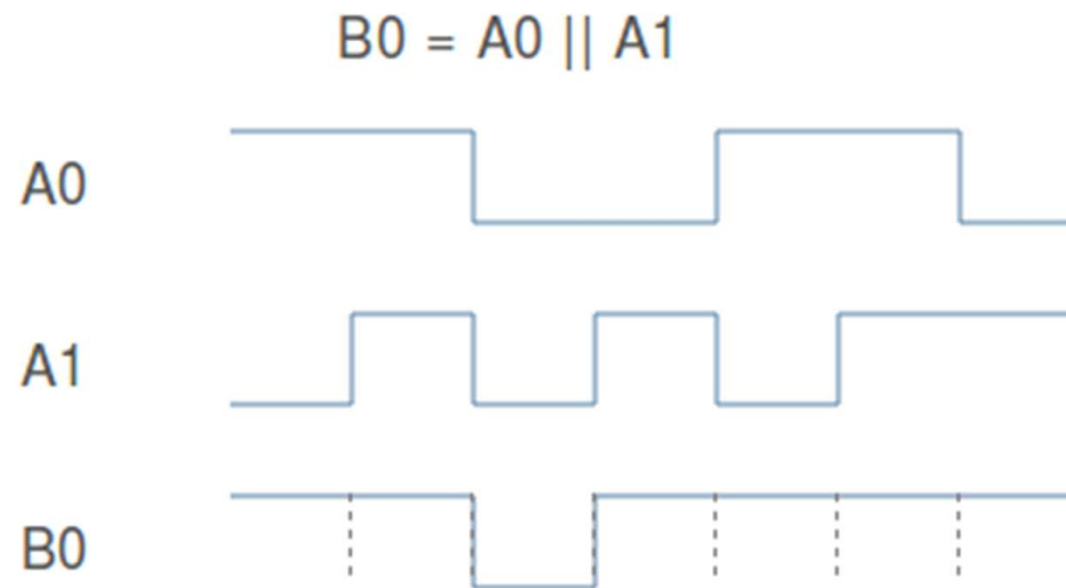
- A0 e A1 tem 4 eventos e B0 tem dois. Se um sinal muda de 0 para 1 é chamado de **subida** de borda, o inverso, de 1 para 0 é **descida** de borda. Um **pulso** é uma porção de sinal iniciada pela subida de um evento e finaliza pela descida de um evento (**corcunda de camelo**).

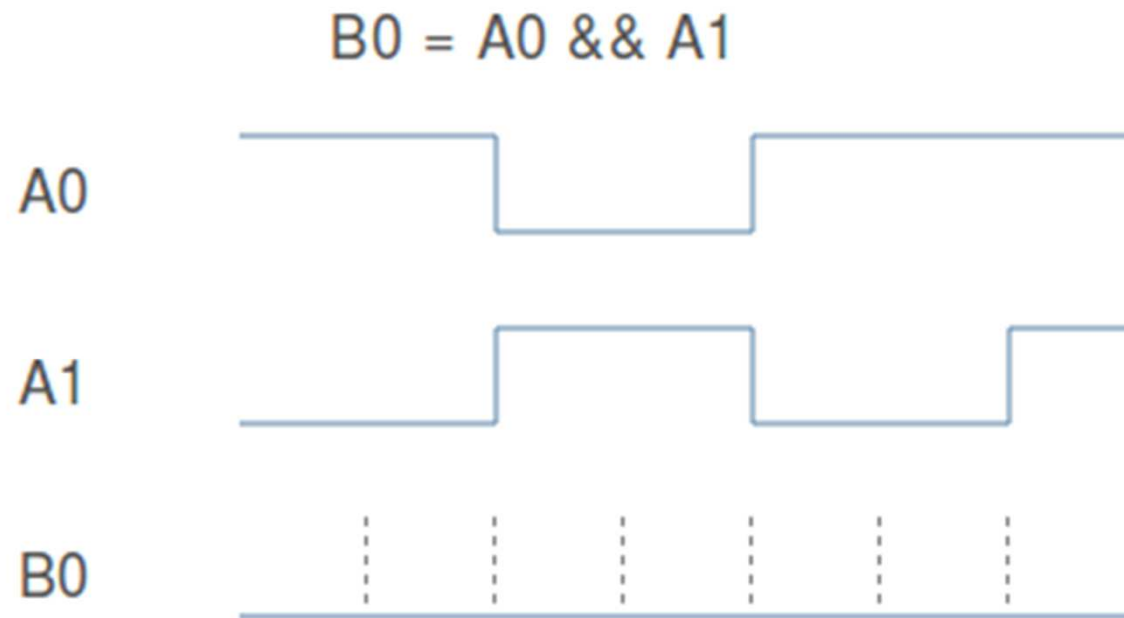
As perguntas refere-se ao diagrama descrito na Figura anterior

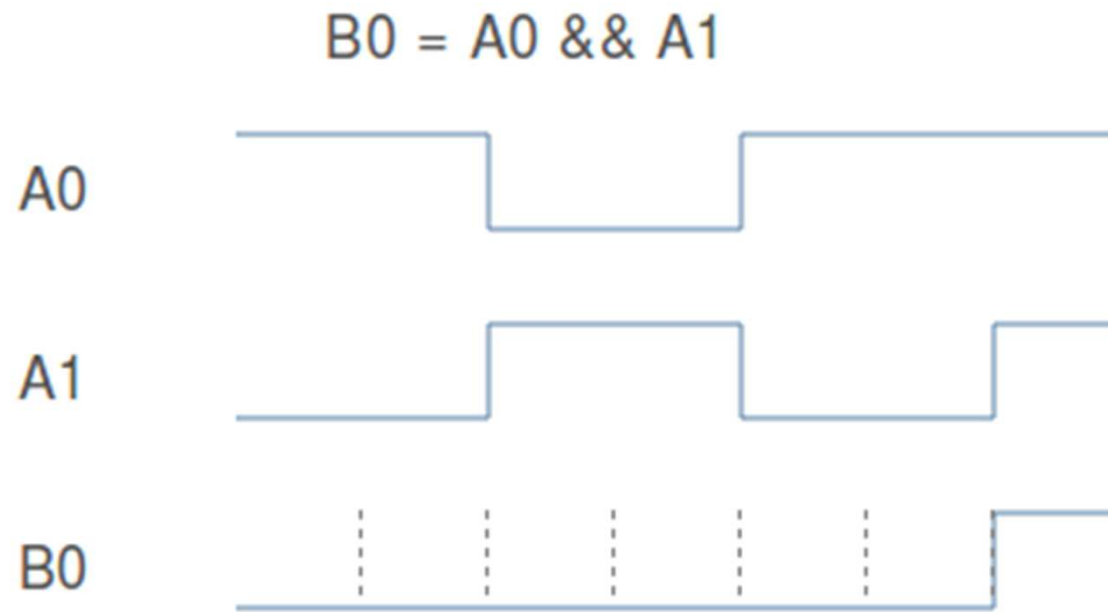
#	Questão	Sua resposta
1	A0 é 1 no tempo 1,5 s.	Verdadeiro
		Falso
2	A0 e A1 são ambos 1 por apenas 0,5 s.	Verdadeiro
		Falso
3	A linha vertical indica que B0 mudando para 1 causa A1 mudar para 1.	Verdadeiro
		Falso
4	B0 exibe três eventos: 0, 1 e 0.	Verdadeiro
		Falso
5	A0 exibe 5 pulsos: 0, então 1, 0, 1 e finalmente 0 novamente.	Verdadeiro
		Falso

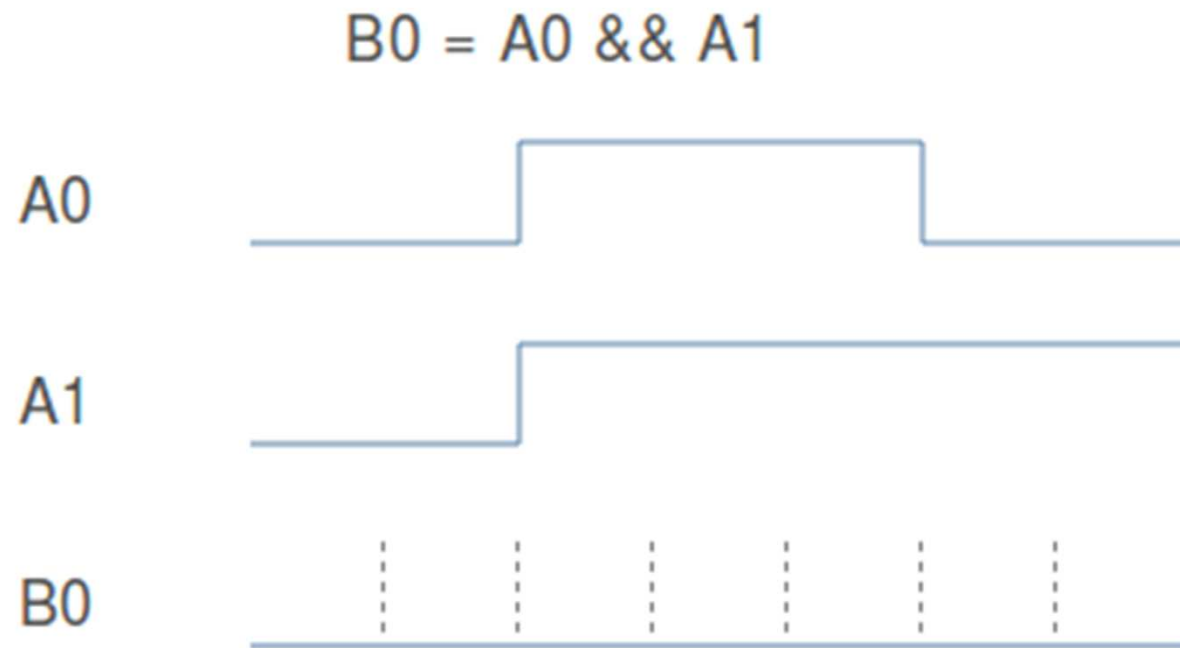
- Ajuste a saída B0 de acordo com as equações e as entradas.



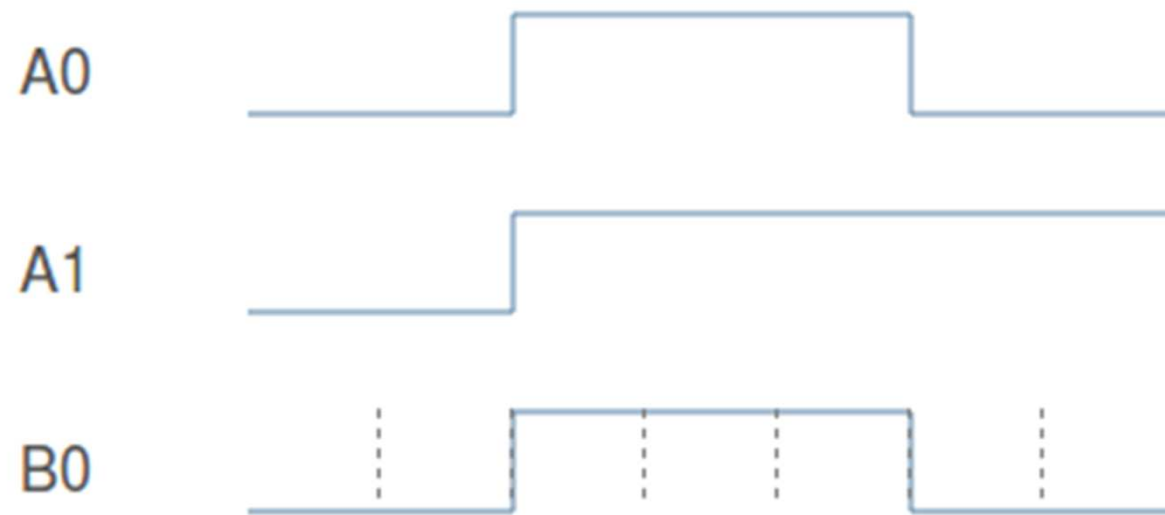


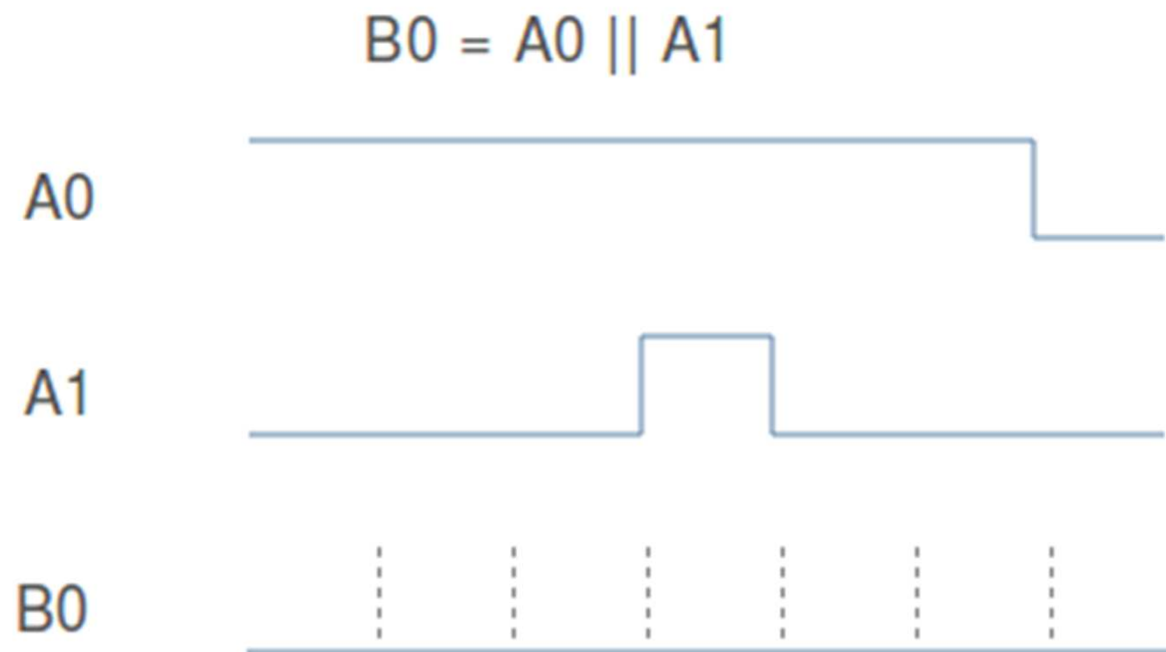


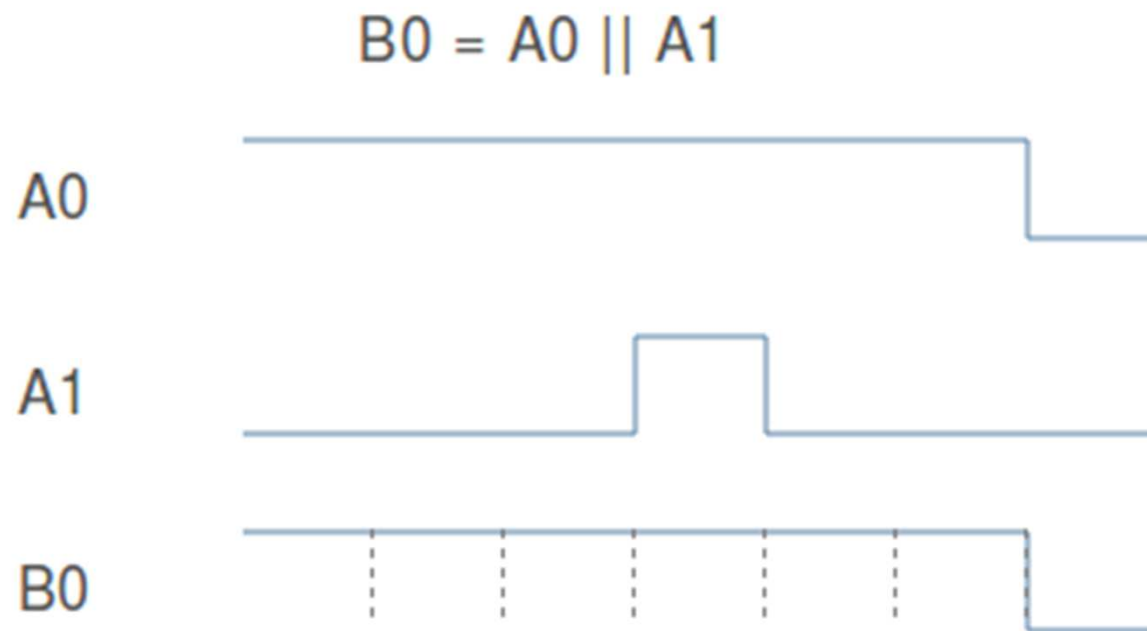




$$B0 = A0 \ \&\& \ A1$$







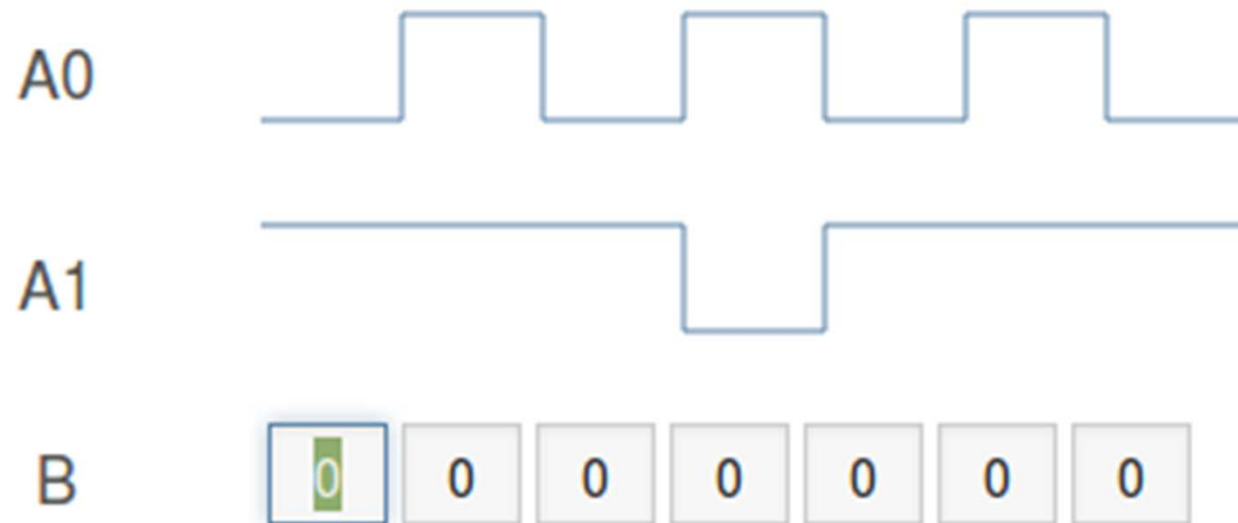
When A0 rises, $B = B + 1$



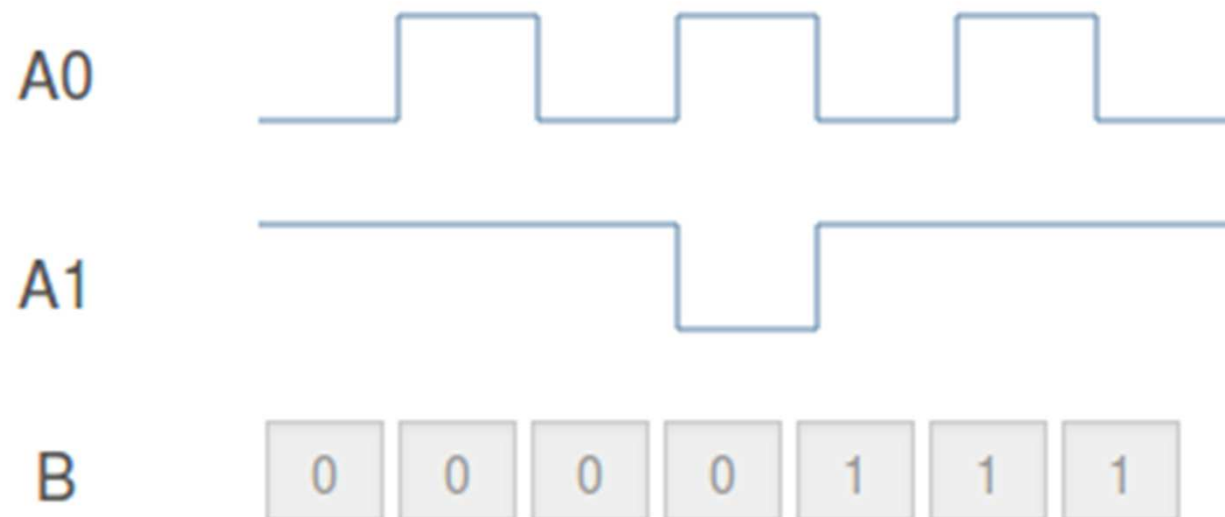
When A0 rises, $B = B + 1$



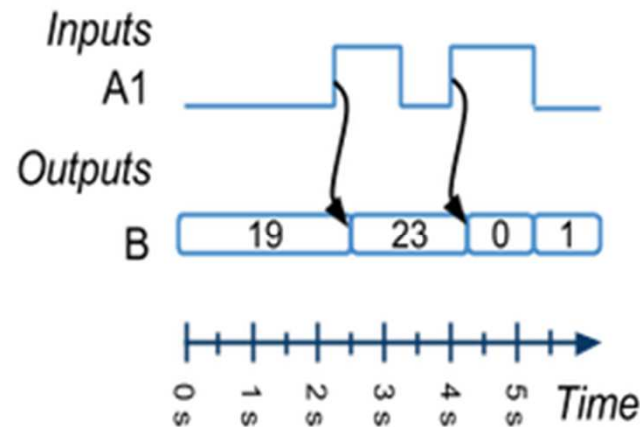
When A0 falls and A1 rises simultaneously, $B = B + 1$



When A0 falls and A1 rises simultaneously, $B = B + 1$



- Diagramas de Tempo mostra valores numéricos, escrevendo-os em retângulos;



- Na figura o valor de B é 19 pelos primeiros 2,5 segundos, então 23 e então 0;
- Também podem incluir setas para indicar uma entrada particular de um evento **disparado** (**triggered**) ou causado alguma mudança na saída (A1 disparou uma mudança na saída B)

As perguntas refere-se ao diagrama descrito na Figura anterior

#	Questão	Resposta
1	O diagrama de tempo indica que brevemente depois do tempo de 2 s um evento de subida em A1 causou B mudar de 19 para 23	Verdadeiro
		Falso
2	O diagrama de tempo indica que por volta de 4 s uma mudança em B de 23 para 0 disparou a subida de A1	Verdadeiro
		Falso
3	O diagrama de evento indica que apenas depois do tempo 5 s o evento de descida em A1 causou B mudar de 0 para 1	Verdadeiro
		Falso
4	Durante o tempo que B é 0 ou 1, o diagrama de tempo poderia mostrar o sinal como alto ou baixo assim como foi feito em A	Verdadeiro
		Falso

Sistemas Embarcados

TESTE DE SISTEMA

- Um código escrito deve ser testado para garantir a sua corretude;
- Um método é gerar diferentes valores de entradas e então observar se os valores de saída estão corretos;
- Para testar a implementação de um código, verifique o seguinte exemplo:

$$B0 = A0 \ \&\& \ !A1$$

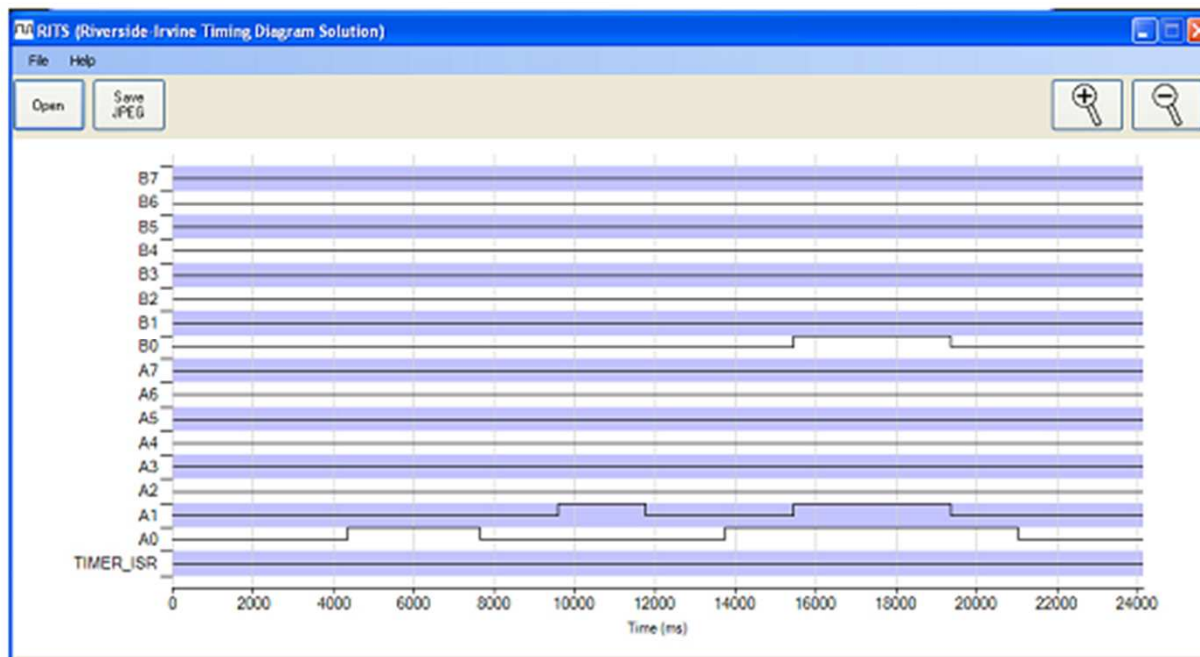
- Todas os possíveis valores de entrada combinando A0 e A1 podem ser: 00, 01, 10, 11;
- Testar com duas combinações é simples, mas se for:

$B0 = A0 \ \&\& \ A1 \ \&\& \ A2 \ \&\& \ A3 \ \&\& \ A4 \ \&\& \ A5 \ \&\& \ A6 \ \&\& \ A7$

- São 256 combinações e se o sistema tiver 32 entradas de 1 bit? Mais de 4 bilhões!!!!

- Por causa que os testes usualmente não cobrem todas as possíveis combinações, estes devem cobrir o que chamamos de casos de bordas (**border cases**);
- Em que os casos extremos, como todas as entradas serem 0 ou todas serem 1 e então vários casos normais;
- Exemplo, testes podem ser em duas bordas A7,...,A0 inicializados com 00000000 (saída deve ser 0) e 11111111 (saída deve ser 1). Outros testes podem ser incluídos como 00110101 ou 10101110 e assim vai...
- Se o código tem brechas, então os testes devem passar por cada estado possível, nesse caso é chamado de **código 100% coberto**.

- A ferramenta RIM pode gerar um diagrama de tempo;
- Para isso execute um código, teste possíveis soluções e em **Tools** → **Generate Timing Diagram**;
- A ferramenta RITS será acionada e mostrada como na figura.



- Salve, compile e execute o programa abaixo. Clique nos tipos de entrada switch para que tenha os seguintes valores A1A0=00, então 00, 10, 00, 01, 11, 00. Aperte o botão "Break" e gere o diagrama de tempo e observe o comportamento.

```
#include "RIMS.h"

void main()
{
    while (1) {
        B0 = A1 && A0;
    }
}
```

- Se um programa falhar, um bom passo de teste é primeiro o programador fazer os passos mentalmente ou no papel e lápis (**rastreio manual**);
- Outra é usar um **rastreio de estado**, que imprime informações sobre a execução de um programa;
 - `printf("Estou aqui"\n);`
- Exemplo: Inclua no exemplo anterior a seguinte linha após a linha de instrução
`printf("B = %d\r\n", B)`

- Um carro tem um sensor conectado a A0 (1 significa que o carro está ligado), outro sensor conectado em A1 (1 significa que uma pessoa está sentada no banco do motorista) e por fim um sensor está conectado em A2 (1 significa que o cinto de segurança está colocado). Escreva um código C no RIM para um sistema de verificação do cinto de segurança, que ilumina uma luz no painel ($B0 = 1$) quando um carro é ligado, um motorista está sentado e o cinto não está apertado. Teste o código escrito no RIM para todas as possíveis combinações A2, A1 e A0 e gere o diagrama de tempo.

- Combinações de valores de entrada conhecidos como vetor de teste podem ser descrito na ferramenta;
- Apenas se cada combinação de valores de entrada sejam gerados por um switch;
- Clique no botão “**Use test vectors**” para ativar o painel de vetor de testes;
- O usuário pode descrever uma sequencia de eventos em ordem.

- Os possíveis eventos podem ser:
 - **Set input** – inicializar a entrada A7-A0 para um valor específico;
 - b00000010
 - 0x02
 - **Wait** – espera por uma específica quantidade de tempo
 - wait 15 ms
 - wait 3 s
 - **Check output** – declara que uma saída B combina com um valor específico
 - assert b11110000
 - assert 0xf0

- O evento check output é também conhecido como declaração (assertion);
- Uma declaração compara as saídas B7-B0 de um dado valor esperado e imprime um "warning" se aqueles valores não batem;
- Afirmações de declarações provê um mecanismo para detectar se um programa não tem o comportamento esperado;
- Todos os valores podem ser expressos em:
 - Binário (b00000111)
 - Hexadecimal (0x07)
 - Decimal (7)

- Abaixo segue um exemplo simples que pode ser usado como um programa de teste que executa o estado $B0 = A0 \ \&\& \ !A1$.

```
b00000000  
wait 100 ms  
assert b00000000  
b00000001  
wait 100 ms  
assert b00000001
```

- O primeiro vetor de teste inicializa as entradas $A1A0$ para 00, espera 100 ms e verifica se $B0$ é 0. O segundo vetor então inicializa $A1A0$ para 01 e declara que $B0$ é 1.

- Use o RIM para testar o programa $B0 = A0 \ \&\& \ !A1$. Copie o vetor de teste anterior e então adicione eventos para verificar o comportamento quando $A1A0$ é 10 ou 11

#	Questão	Resposta
1	Um requisito de teste mínimo é testar todas as possíveis combinações de entradas de um sistema	Verdadeiro
		Falso
2	Um vetor de teste é uma combinação particular de valores de entrada	Verdadeiro
		Falso
3	Um caso de borda é uma típica entrada de valores para um programa	Verdadeiro
		Falso
4	Um rastreo de estado imprime informações sobre a execução de um programa	Verdadeiro
		Falso



Sistemas Embarcados - Introdução

Instituto Federal de Pernambuco
Coordenação de Informática/IFPE

Anderson L. S. Moreira
`anderson.moreira@recife.ifpe.edu.br`

