SBL0082 - Microprocessadores
Aula 02 - Arquiteturas Básicas

Prof. Me. Alan Marques da Rocha

Universidade Federal do Ceará (*Campus* Sobral)
Bacharelado em Engenharia Elétrica e Computação

18 de setembro de 2025

Roteiro da Aula

Objetivos da Aula

- Diferenciar **microprocessador** e **microcontrolador** no contexto de sistemas embarcados.
- Entender as ideias centrais do computador com um conjunto reduzido de instruções ou *Deduced Instruction Set Computer* (**RISC**).

- Entender as ideias centrais do computador com um conjunto complexo de instruções ou *Complex Instruction Set Computer* (**CISC**), com vantagens e limitações.
- Compreender os três barramentos essenciais: **dados**, **endereços** e **controle**.
- Comparar as arquiteturas **Von Neumann** e **Harvard**, incluindo impactos em desempenho.

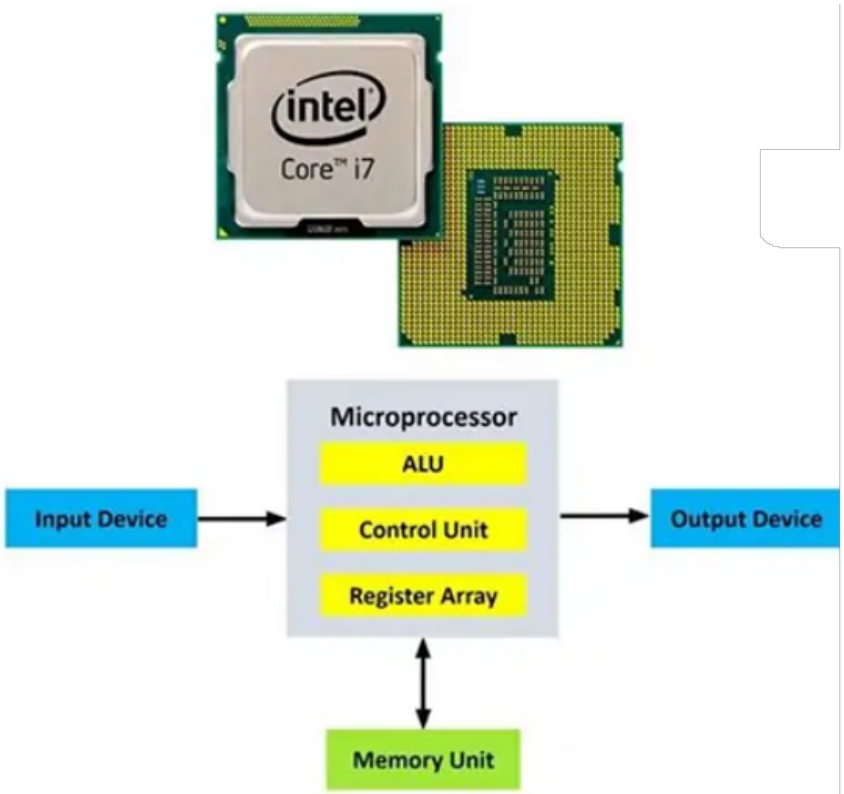
Contexto no Componente Curricular

Esta aula dá continuidade à visão sucinta da teoria e prepara o terreno para:

- Memórias, periféricos.
- Dispositivos microcontrolados.

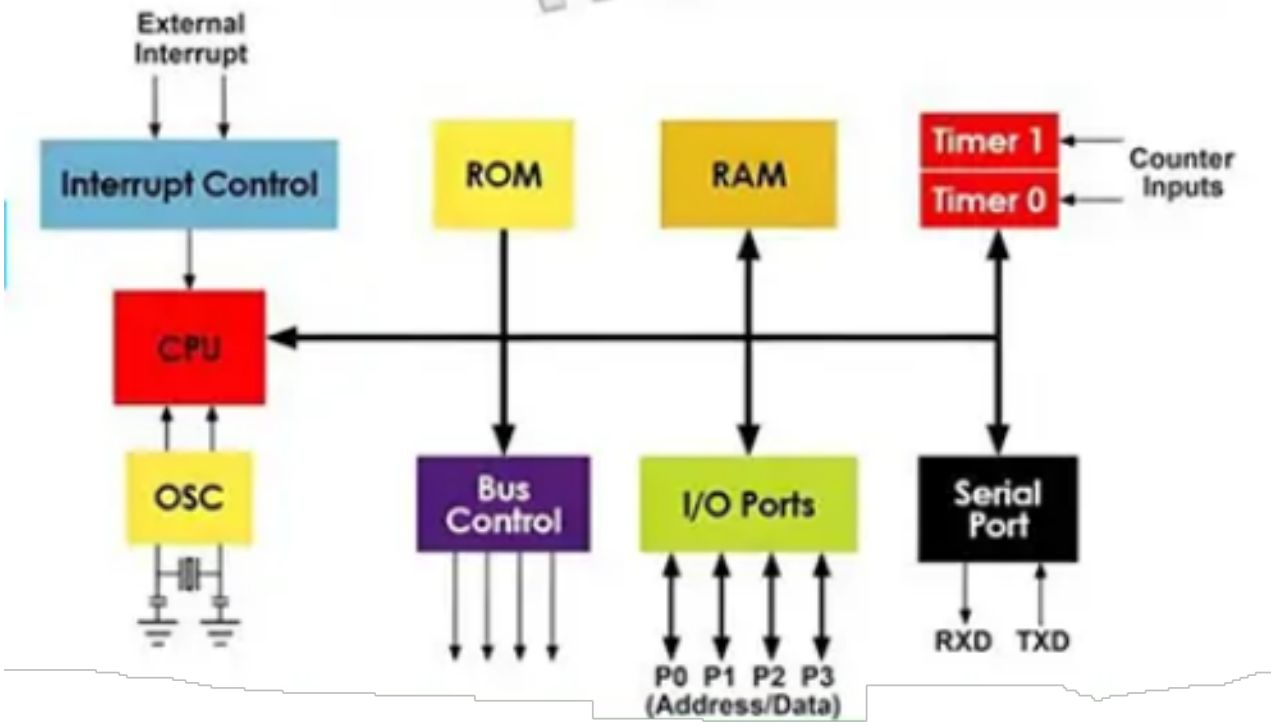
Microprocessador vs Microcontrolador

Microprocessador: CPU em um único CI, composta por unidade de controle, ULA e registradores. Necessita de memória e E/S externos para formar um sistema mínimo.

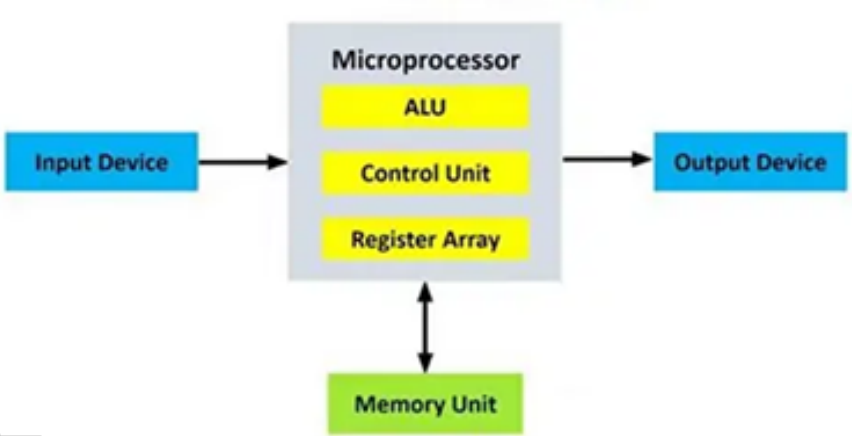


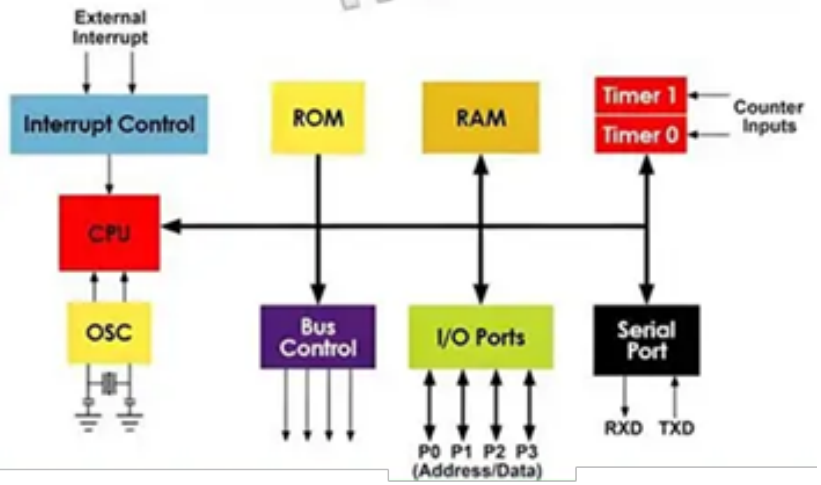
Microcontrolador: Computador completo em um único CI, usualmente com RAM, memória de programa, temporizadores, portas seriais/paralelas, conversores A/D e controladores de interrupção. Focado em aplicações específicas.


MICROCHIP
ATmega328P


External Interrupt
Interrupt Control
ROM
RAM
Timer 1
Timer 0
Counter Inputs
CPU
OSC
Bus Control
I/O Ports
Serial Port
RXD
TXD
P0 P1 P2 P3
(Address/Data)

Microprocessor vs Microcontroller





Bits, Bytes e Palavras

- **Bit**: *Binary digit* ou dígito binário que assume 0 ou 1.
- **Byte**: 8 bits; unidade básica de dados.
- **Nibble/Word/Dword**: 4/16/32 bits, respectivamente.
- Largura de palavra, barramentos e registradores influenciam desempenho e faixa de endereçamento.

Exercício Básico

→ **(Desafio em grupo):**

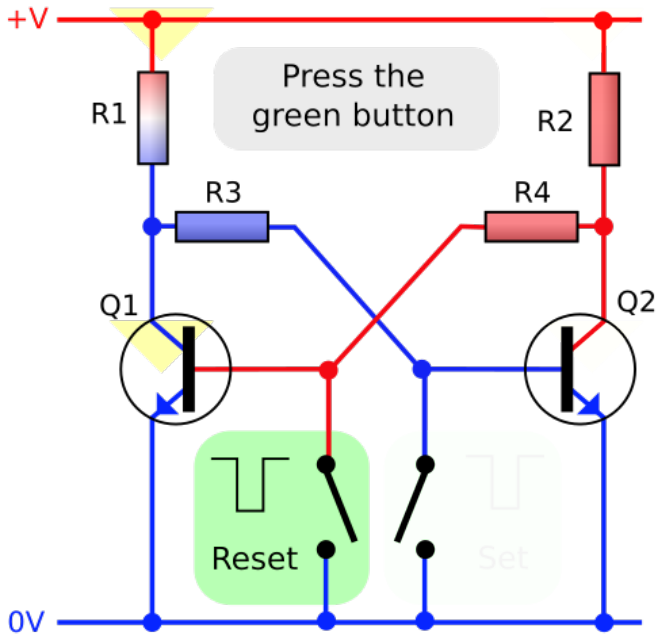
Cada equipe deve converter uma frase secreta enviada em código binário (cada letra em ASCII binário) e descobrir a mensagem. A equipe adversária será desafiada a "decodificar" a mensagem relacionada a eles.

Exemplo: `"01100011 01101111 01100100 01100101"` → `"code"`.

Código Padrão Americano para o Intercâmbio de Informação (ASCII):

Flip-Flops e Flags

Flip-flop (FF): elemento sequencial que armazena 1 bit, base de contadores, registradores e memórias.



Quatro tipos de FF possuem aplicações comuns em sistemas de *clock* não-sequencial. Quais são eles?

Na literatura de eletrônica digital, os FFs são elementos **sequenciais** usados em **sistemas com *clock*** (síncronos) e também em arranjos **assíncronos**. Assim, em vez de "sistemas de *clock* não-sequencial", adotaremos: **"aplicações em sistemas síncronos e assíncronos"**.

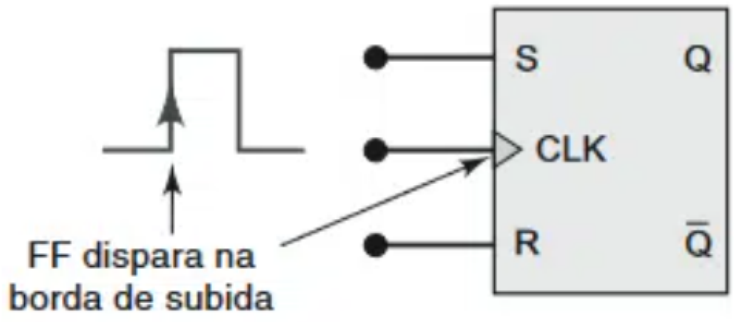
- **S–R** (*set–reset*): força Q a 1 (set) ou a 0 (reset); em versões puras, a condição $S = R = 1$ é indeterminada.

- **S–R** (*set–reset*): força Q a 1 (set) ou a 0 (reset); em versões puras, a condição $S = R = 1$ é indeterminada.
- **J–K**: generaliza o S–R e resolve a condição indeterminada; com $J = K = 1$, ocorre *toggle*.

- **S–R** (*set–reset*): força Q a 1 (set) ou a 0 (reset); em versões puras, a condição $S = R = 1$ é indeterminada.
- **J–K**: generaliza o S–R e resolve a condição indeterminada; com $J = K = 1$, ocorre *toggle*.
- **D** (*data*): amostra o dado D na borda do clock: "guarda o que chega".

- **S–R** (*set–reset*): força Q a 1 (set) ou a 0 (reset); em versões puras, a condição $S = R = 1$ é indeterminada.
- **J–K**: generaliza o S–R e resolve a condição indeterminada; com $J = K = 1$, ocorre *toggle*.
- **D** (*data*): amostra o dado D na borda do clock: "guarda o que chega".
- **T** (*toggle*): alterna o estado quando $T = 1$; mantém quando $T = 0$.

Flip-Flop S–R (Set–Reset):



Entradas			Saída
S	R	CLK	Q
0	0	↑	Q_0 (não muda)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambíguo

Q_0 é o nível de saída anterior a ↑ de CLK.
↓ de CLK não produz mudança em Q.

Equação característica (versão síncrona idealizada):

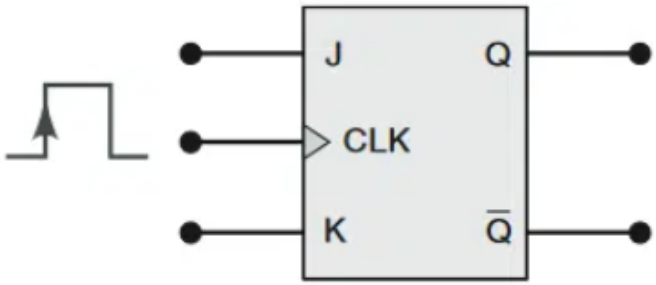
$$Q_{t+1} = S \vee (\overline{R} \wedge Q_t)$$

Interpretação: se $S = 1$ então $Q_{t+1} = 1$; se $R = 1$ então $Q_{t+1} = 0$; caso contrário, mantém Q_t.

Observação: no *latch* S–R assíncrono, a condição $S = R = 1$ é *proibida*. Em flip-flops síncronos comerciais, o circuito interno evita estados inválidos.

Variáveis: Q_t é o estado atual; Q_{t+1} é o próximo estado; $\bar{\cdot}$ denota negação; $\vee$ é OR; $\wedge$ é AND.

Flip-Flop J–K:



J	K	CLK	Q
0	0	↑	Q_0 (não muda)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q_0}$ (comuta)

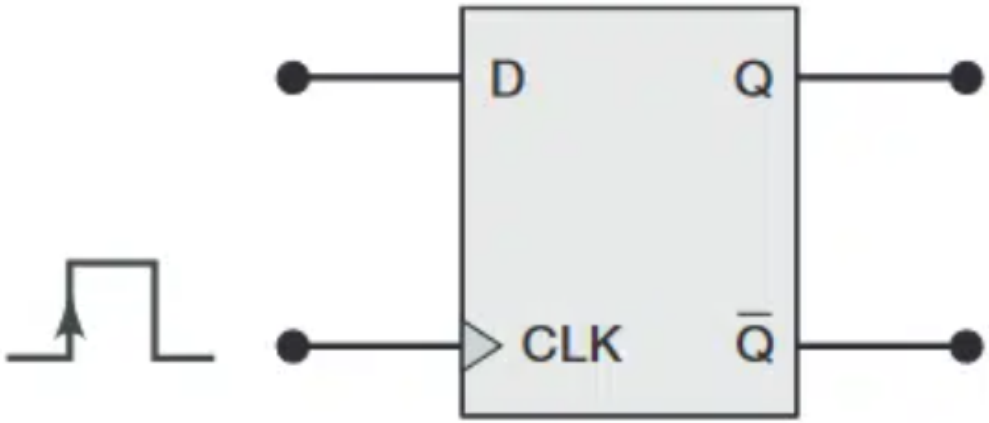
Equação característica:

$$Q_{t+1} = J\,\overline{Q_t} \;\vee\; \overline{K}\,Q_t$$

Casos notáveis: $J = K = 0 \Rightarrow Q_{t+1} = Q_t$ (mantém); $J = 1, K = 0 \Rightarrow Q_{t+1} = 1$ (set); $J = 0, K = 1 \Rightarrow Q_{t+1} = 0$ (reset); $J = K = 1 \Rightarrow Q_{t+1} = \overline{Q_t}$ (toggle).

Uso típico: base para contadores síncronos, divisores e lógica de controle onde o *toggle* é desejado.

Flip-Flop D (Data):



D	CLK	Q
0	↑	0
1	↑	1

Equação característica:

$$Q_{t+1} = D$$

Sentido físico: o dado de entrada é amostrado na borda ativa do clock e transferido para Q.

Uso típico: registradores, *pipelines*, registradores de deslocamento, sincronizadores entre domínios de clock.

Flip-Flop T (Toggle):

Equação característica:

$$Q_{t+1} = T \oplus Q_t$$

Casos: se $T = 0$, mantém Q_t; se $T = 1$, alterna (*toggle*).

Uso típico: divisores de frequência por 2, controle de duty-cycle, contadores binários quando encadeado.

Símbolos: $\oplus$ é XOR.

Comparativo Rápido:

Tipo	Equação	Comportamento resumido
S–R	$Q_{t+1} = S \vee (\overline{R}\, Q_t)$	Set/Reset/Manter; versão pura tem caso proibido ($S = R = 1$).
J–K	$Q_{t+1} = J\overline{Q_t} \vee \overline{K}\, Q_t$	Generaliza S–R; com $J = K = 1$ faz *toggle*.
D	$Q_{t+1} = D$	Amostra o dado; ideal para registradores/pipeline.
T	$Q_{t+1} = T \oplus Q_t$	Alterna quando $T = 1$; divisor por 2 natural.

Aplicações Comuns:

1) Divisor de frequência: com FF T (ou J–K com $J = K = 1$), cada FF divide a frequência por 2; cadeias dividem por 2^n.

2) Contadores binários: assíncronos (*ripple*) com T/JK; síncronos com JK/D para maior velocidade e controle de estados.

3) Registradores/Pipeline: FF D como memória de estágio em *pipelines* de processadores, controladores e DSPs.

4) *Debouncing* de teclas: *latch* S–R (ou rede RC + Schmitt) para eliminar *bounce*, seguido de FF D para sincronizar ao clock.

5) Detectores de borda: dois FF D em série e uma XOR para gerar pulso estreito na subida/descida do sinal observado.

6) Sincronização entre domínios de clock (CDC): cadeia de 2–3 FF D para reduzir probabilidade de metastabilidade ao cruzar domínios.

7) Máquinas de estados finitos (FSM): bancos de FF D (ou JK) armazenam o estado; a lógica combinacional determina D (ou J, K) a cada ciclo.

Flags: bits de estado (ex.: Sinal, Zero, Carry, Paridade, Auxiliar de Carry) que registram o resultado de operações e orientam desvios condicionais em código Assembly.

Ideia Central

CISC: prioriza instruções complexas e ricas, tipicamente com múltiplos ciclos por instrução.

RISC: prioriza poucas instruções simples, com execução rápida, favorecendo pipeline e compilação eficiente.

CISC: Características

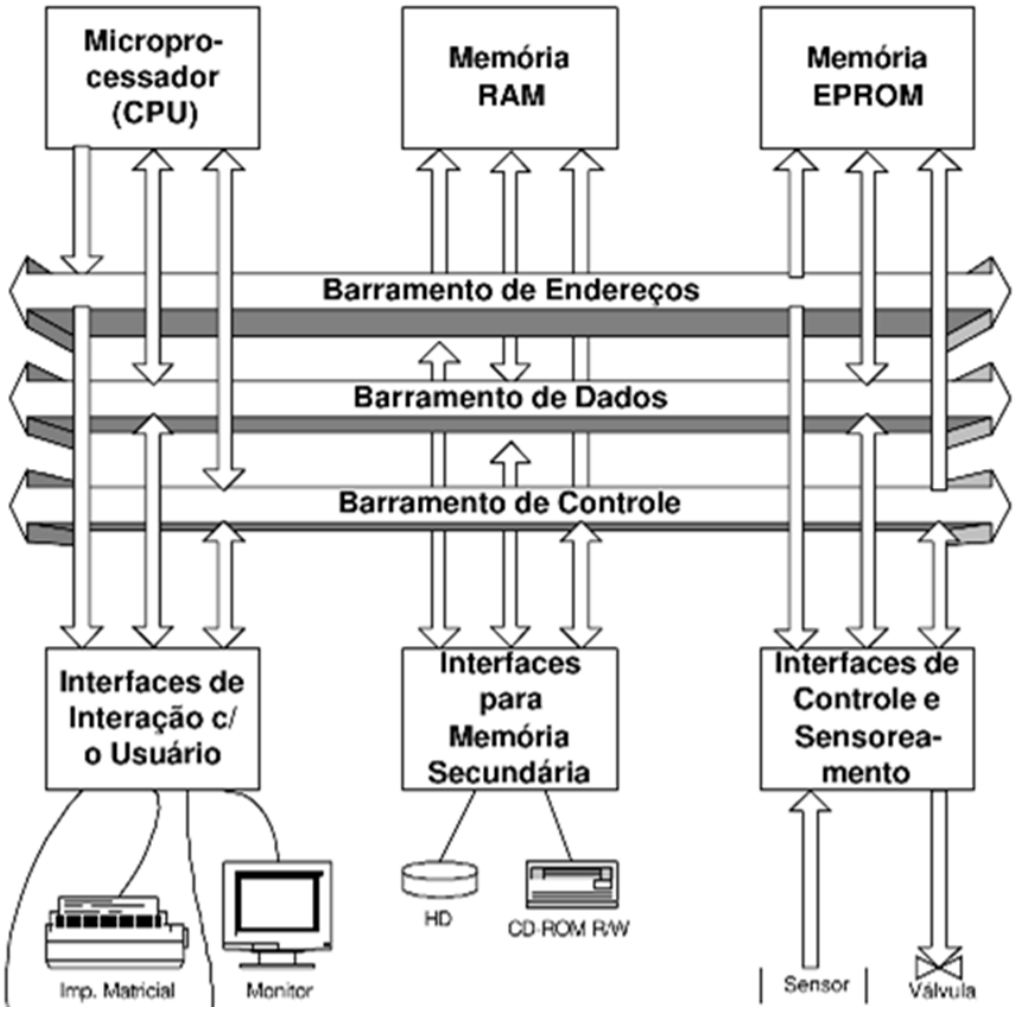
- Conjuntos de instruções amplos e com modos de endereçamento diversos.
- Frequente execução em vários ciclos por instrução.
- Menos registradores de propósito geral em designs clássicos.
- Exemplos históricos: x86 clássicos de desktop e servidores.

RISC: Características

- Conjunto reduzido de instruções, formato regular e decodificação simples.
- Forte uso de registradores de propósito geral.
- Pipeline otimizado e alta taxa de *throughput*.
- Exemplos: ARM, MIPS, RISC-V e clássicos como IBM 801, Berkeley RISC.

Comentários Atuais

Implementações modernas frequentemente mesclam ideias: processadores x86 decodificam instruções complexas em micro-ops internas mais simples, aproximando-se de um fluxo *RISC-like*. Em contrapartida, ISAs RISC evoluíram com extensões vetoriais e de ponto flutuante, mantendo simplicidade no núcleo.


Micropro-
cessador
(CPU)
Memória
RAM
Memória
EPROM
Barramento de Endereços
Barramento de Dados
Barramento de Controle
Interfaces de
Interação c/
o Usuário
Interfaces
para
Memória
Secundária
Interfaces de
Controle e
Sensorea-
mento
Imp. Matricial
Monitor
HD
CD-ROM R/W
Sensor
Válvula

Visão Geral

Um sistema de barramentos é o conjunto de linhas de sinal que interconecta **CPU, memórias** e **periféricos**. Em microcomputadores clássicos consideramos:

Barramento de Endereços

Endereços: unidirecional a partir da CPU, seleciona a "casa" de memória ou o dispositivo de E/S alvo.

Observação: largura do barramento de endereços define o espaço endereçável (2^n posições para n linhas).

Barramento de Dados

Dados: bidirecional, transporta palavras entre CPU, memórias e periféricos.

Impacto: largura do barramento de dados afeta quantos bits trafegam por ciclo, influenciando desempenho efetivo.

Barramento de Controle

Controle: sinais como leitura/escrita de memória, entrada/saída, interrupções e confirmação de ciclo. Coordena o momento de transferência e a direção do fluxo.

Von Neumann: Conceito

Memória única e **barramento compartilhado** para instruções e dados. Simplicidade arquitetural, mas sujeito ao gargalo do mesmo caminho para buscar instruções e acessar dados.

Von Neumann: Implicações

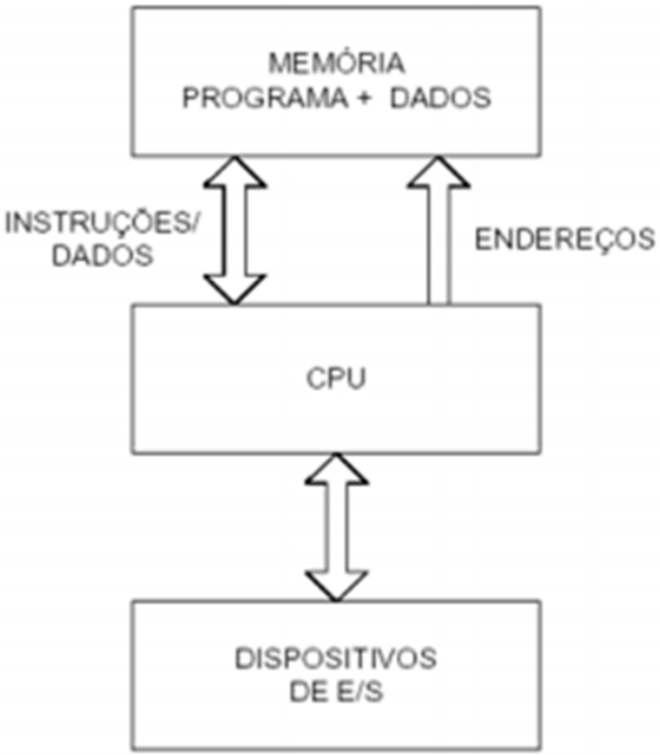
- Código de instruções potencialmente mais complexo (histórico CISC).
- Vários ciclos por instrução são comuns em designs clássicos.
- Menos portas lógicas, mas velocidade limitada pelo compartilhamento do barramento.

Harvard: Conceito

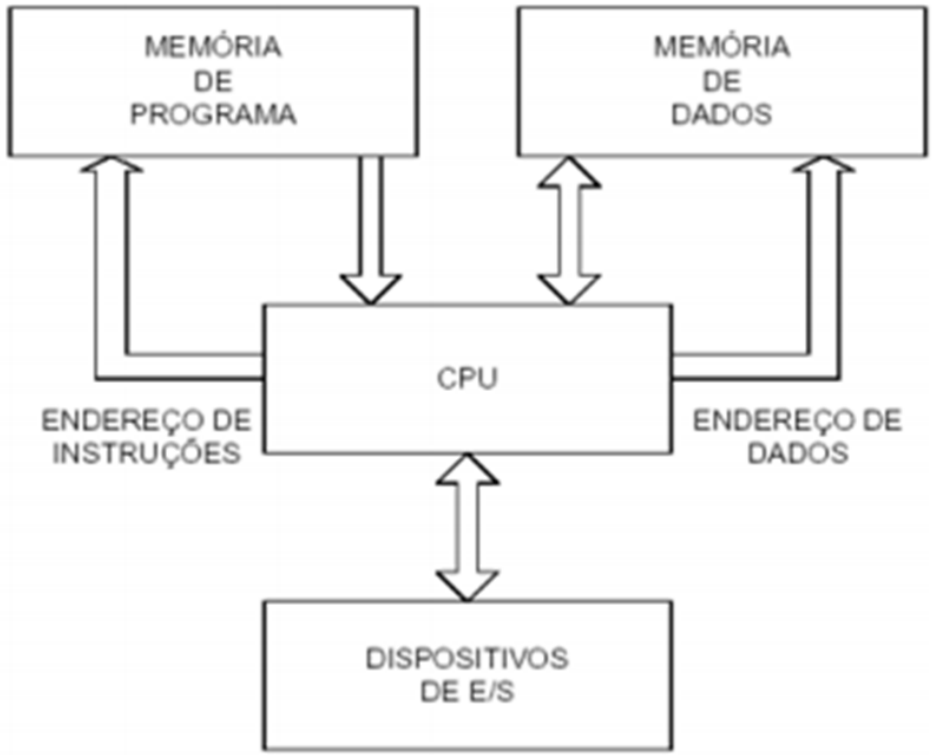
Memórias e barramentos **separados** para instruções e dados. Permite buscar instruções enquanto dados são lidos/escritos, favorecendo paralelismo estrutural.

Harvard: Implicações

- Conjuntos de instruções *enxutos* e *pipeline* eficiente.
- Possibilidade de executar uma instrução por ciclo em projetos bem otimizados.
- Muito comum em microcontroladores (ex.: famílias PIC e ARM Cortex-M).



Arquitetura
Von-Neumann



Arquitetura
Harvard

Atividade:

Façam um resumo apresentando os conceitos básicos sobre cada arquitetura, diferenças de hardware, exemplos, aplicações práticas e outras informações relevantes.

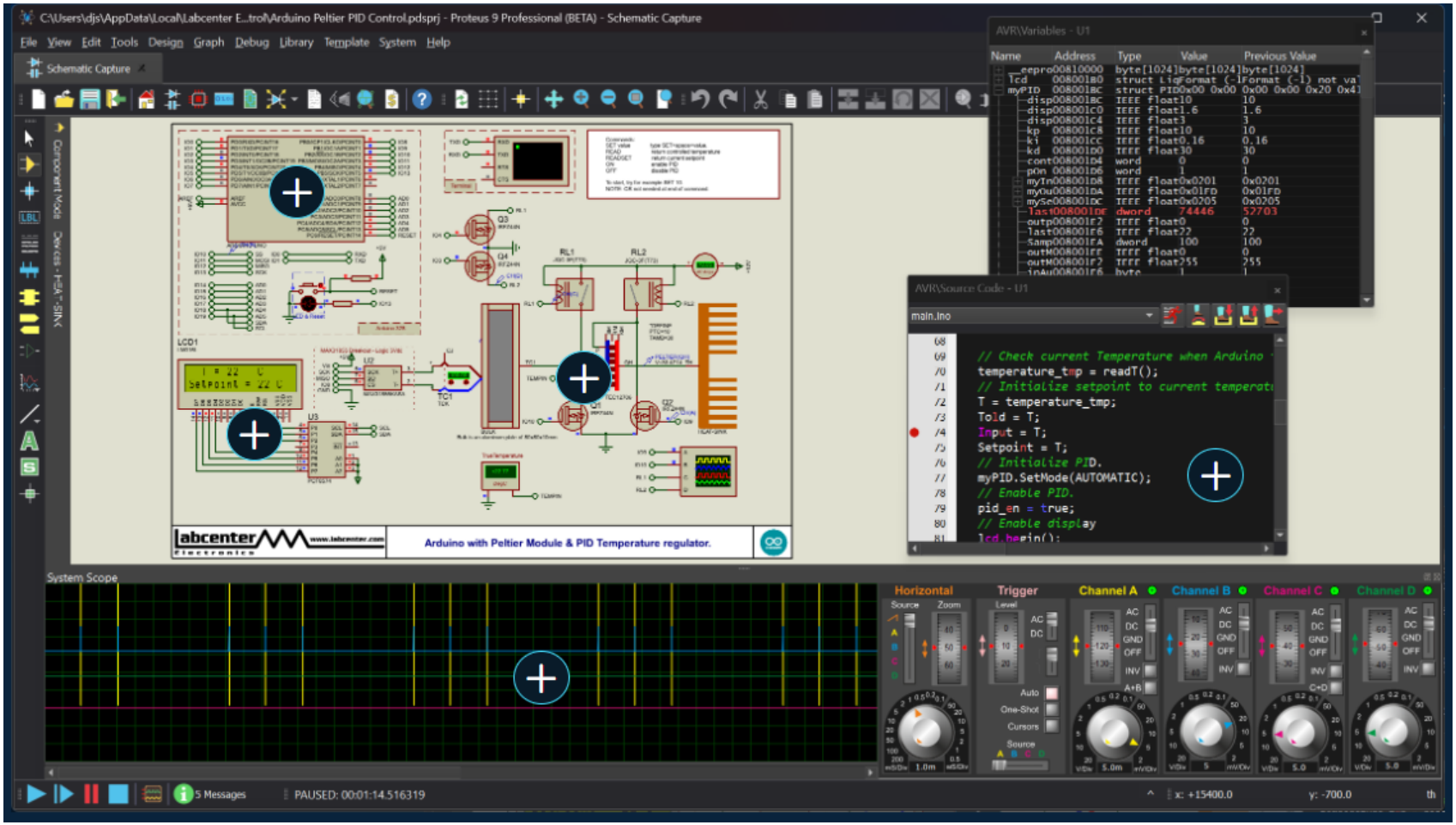
- O resumo deverá ser feito à mão (escrito), escaneado e enviado em formato PDF via tarefe no SIGAA.
- Valerá como parte do "choro" para a **nota final** da disciplina.
- **NÃO É OBRIGATÓRIO. FAZ QUEM QUER!**

Resumo da Aula

- Revisamos conceitos fundamentais (bit/byte, flip-flops, flags).
- Diferenciamos RISC e CISC com foco em implicações práticas.
- Apresentamos os três barramentos e sua função no sistema.
- Comparamos Von Neumann e Harvard.

Próximos Passos

- Disponibilizar o *software* simulador do microcontrolador e o passo a passo de instalação.


C:\Users\djs\AppData\Local\Labcenter E...trol\Arduino Peltier PID Control.pdsprj - Proteus 9 Professional (BETA) - Schematic Capture
File View Edit Tools Design Graph Debug Library Template System Help
Schematic Capture
AVR\Variables - U1
Name Address Type Value Previous Value
AVR\Source Code - U1
main.ino
// Check current Temperature when Arduino
temperature_tmp = readT();
// Initialize setpoint to current temperatu
T = temperature_tmp;
Told = T;
Input = T;
Setpoint = T;
// Initialize PID.
myPID.SetMode(AUTOMATIC);
// Enable PID.
pid_en = true;
// Enable display
lcd.begin();
labcenter Electronics
Arduino with Peltier Module & PID Temperature regulator.
System Scope
Horizontal
Trigger
Channel A
Channel B
Channel C
Channel D
5 Messages
PAUSED: 00:01:14.516319

Para a Próxima Aula

- Introdução a memórias: voláteis e não voláteis; células básicas.
- Tipos: ROM, EPROM, EEPROM, OTP, SRAM, DRAM.
- Leitura de trechos do *datasheet* do PIC18F452 sobre organização de memória.

Bibliografia Básica

[1] SOUSA, D. R.; SOUZA, D. J. *Desbravando o Microcontrolador PIC18: Ensino Didático*. São Paulo: Érica, 2012.

[2] SOUZA, D. J.; LAVINIA, N. C.; SOUZA, D. R. *Desbravando o Microcontrolador PIC18: Recursos Avançados*. São Paulo: Érica, 2010.

[3] PEREIRA, F. *Microcontroladores PIC*. São Paulo: Érica, 2009.

Muito Obrigado!!

E-mail: eng.alanmarquesrocha@gmail.com