



UNIVERSIDADE FEDERAL DE SÃO PAULO
Instituto de Ciência e Tecnologia
Engenharia Biomédica

Sistema de Alarme de Incêndio Inteligente Baseado em Cálculo de Risco em MIPS

Discentes: Nicolas David da Cruz Santos , Davi de Oliveira Custódio, João Augusto Paixão Rocha,
Bernardo Friske de Souza

1. Resumo

O presente trabalho apresenta a criação de um simulador de alarme de incêndio com uso da arquitetura MIPS. A partir das métricas de temperatura e fumaça, foi elaborado um sistema que faz um cálculo de risco ponderado que informa como saída uma quantificação do risco e o estado do cenário atual, com este variando entre normal, de atenção, de alerta e de evacuação. Para isso, foram implementadas funções, laços, tratamento de entrada, pilha, lista circular, vetores, entre outros recursos da linguagem assembly na arquitetura MIPS. Foi possível concluir que, a partir da linguagem de baixo nível assembly, programas com aplicações reais podem ser implementados e, com associação a sensores externos e um dispositivo que integre hardware e software, podem ter uso efetivo.

Palavras-chave: Simulador de Alarme de Incêndio. AOC. Hardware. Sistemas Embarcados

2. Introdução

Os sistemas embarcados estão presentes em muitos dispositivos do dia a dia, incluindo sensores e alarmes usados para monitorar ambientes. Para entender melhor como esses sistemas funcionam internamente, é comum usar arquiteturas didáticas como o MIPS, que permite observar de forma clara como operações são feitas diretamente por registradores e instruções de baixo nível (PATTERSON; HENNESSY, 2014).

Neste trabalho, foi feito o desenvolvimento de um simulador de sistema embarcado/IoT implementado em linguagem MIPS Assembly, seguindo as restrições e características de dispositivos de baixo nível. O objetivo do projeto é demonstrar como recursos limitados — como memória reduzida, ausência de bibliotecas de alto nível e necessidade de controle manual da pilha e da lógica interna — influenciam diretamente a construção de sistemas embarcados reais.

O sistema simulado tem como finalidade monitorar temperatura e nível de fumaça, calculando um índice de risco e determinando o estado do ambiente com base nesses parâmetros. O programa também permite ativação manual do alarme e exibição das últimas leituras registradas, implementando assim funcionalidades esperadas em dispositivos de segurança residenciais ou industriais.

3. Descrição Geral do Sistema

3.1 Objetivos do simulador

O objetivo do simulador é mimetizar o comportamento de um sistema de monitoramento anti-incêndio que vigia a temperatura e os níveis de fumaça de um ambiente. O programa deve receber uma entrada do usuário e fornecer uma resposta, como é melhor descrito abaixo.

3.2 Funcionalidades Implementadas

- Leitura simulada de sensor de temperatura.
- Leitura simulada de sensor de fumaça.
- Ativação manual do alarme.
- Listagem das 10 últimas leituras mais recentes feitas (tanto de temperatura quanto de fumaça).
- Cálculo de risco.
- Interface.

3.3 Arquitetura do programa

O programa começa exibindo a instrução principal “Insira (Temperatura Fumaça) separadas por espaço ou insira o código 'b' para ativação manual do alarme, ou o código 'l' para listar as últimas leituras. Digite 'e' para encerrar.”, e então ele aguarda a entrada do usuário. Então, existem 4 situações a partir daqui:

- **O usuário entrou com dois inteiros (Temperatura e nível de fumaça):** nessa primeira situação o programa calcula o risco utilizando a fórmula já citada anteriormente e imprime uma saída para o usuário.
- **O usuário entrou com o caractere “b”:** nessa segunda situação, o alarme é disparado imediatamente com o risco mais alto possível. O programa imprime um aviso dizendo que o alarme foi acionado manualmente.
- **O usuário entrou com o caractere “l”:** nessa situação o programa imprime as últimas leituras feitas, sendo no máximo as últimas 10 leituras. Caso não acha leituras, o programa imprime um erro dizendo que não existe leituras feitas.
- **O usuário entrou com o caractere “e”:** nesse caso o programa é encerrado imediatamente.

Com exceção da última situação, em todos os casos o programa volta a instrução principal para realizar mais leituras. Caso qualquer entrada inesperada seja inserida, o programa imprime um aviso de erro de “leitura nos sensores” e retorna a instrução principal para uma nova leitura.

4. Máquina de Estados

4.1 Estados do sistema e transições

O sistema implementado funciona como uma máquina de estados, onde cada estado do sistema é determinado pelo valor de risco calculado a partir dos dados de temperatura e fumaça fornecidos pelo usuário. Além disso, eventos externos como o comando de ativação manual do alarme também podem alterar os estados, forcing uma transição para o estado de Evacuação.

Os estados principais do sistema são os seguintes:

- Normal: Estado inicial, onde o risco é inferior a 10, indicando que não há perigo iminente. Se o risco ultrapassar 10, o sistema transita para o estado de Atenção.
 - Transição: Se o risco calculado estiver acima de 10, o sistema transita para o estado de Atenção.
- Atenção: O risco está entre 10 e 30, indicando uma situação de vigilância. Se o risco aumenta além de 30, o sistema entra em estado de Alerta.
 - Transição: Se o risco aumentar além de 30, o sistema transita para o estado de Alerta. Se o risco cair abaixo de 10, o sistema retorna ao estado Normal.
- Alerta: O risco está entre 30 e 50, representando uma situação perigosa. Se o risco ultrapassar 50, ou o alarme for ativado manualmente, o sistema entra em Evacuação.
 - Transição: Se o risco ultrapassar 50 ou se o comando de ativação manual do alarme for executado, o sistema transita para o estado de Evacuação. Caso o risco diminua para menos de 30, o sistema retorna ao estado de Atenção.
- Evacuação: O risco é superior a 50, ou o alarme é ativado manualmente pelo usuário. O sistema recomenda evacuação imediata.
 - Transição: Se o risco cair para níveis abaixo de 50, o sistema retorna ao estado de Alerta.

4.2 Diagrama de Estados

O diagrama a seguir foi feito com o site: <https://dreampuf.github.io/GraphvizOnline>

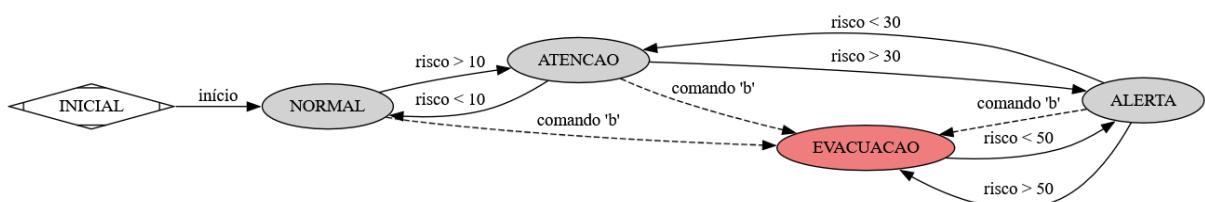


Figura 1: Diagrama de Estados

5. Estruturas de Dados

5.1 Pilhas e Vetores

No projeto, uma pilha é utilizada para guardar valores temporários, organizar chamadas de funções e manter o estado do programa. A manipulação direta do ponteiro de pilha (usando \$s0) mostra como variáveis locais e endereços de retorno são armazenados e restaurados durante a execução, reforçando os princípios de alocação dinâmica e escopo (PATTERSON; HENNESSY, 2005).

O projeto também faz uso de vetores, que são armazenados no segmento .data e acessados por meio de endereçamento baseado em deslocamentos. O uso de indexação com multiplicação por 4 (devido ao tamanho de 4 bytes das palavras) apresenta conceitos de acesso sequencial, aritmética de ponteiros e manipulação de estruturas lineares. Esse tipo de operação é fundamental para entender como a memória é organizada. No projeto, foram utilizados dois vetores para manter o histórico das últimas 10 leituras. Os vetores “UltimasLeiturasT” e “UltimasLeiturasF” são responsáveis por guardar o histórico das leituras de temperatura e fumaça, respectivamente.

5.2 Representação de Sensores e Atuadores

As representações de sensores no projeto são feitas por valores inteiros. Esses valores são obtidos através de leitura via entradas do usuário, ou seja, as informações de temperatura e fumaça são imputadas pelo usuário, e depois entram no histórico pelos vetores.

O atuador no projeto é o alarme. O alarme é representado pelas mensagens de texto na saída, que avisam caso esse alarme esteja ativo. A ativação pode ser tanto automática, caso o cálculo de risco entregue um valor dito como perigoso, ou, manualmente, através de uma entrada específica, mostrada nas instruções do programa.

5.3 Organização de Dados na Memória

A memória, no segmento .data, é utilizada para gerar buffers, pilhas, vetores, entre outros. O buffer criado em “buffer: .space 32” serve como buffer para as entradas do usuário. E a pilha, gerada em “PilhaGeral: .space 32” tem como função inverter a entrada do usuário, a fim de a entrada poder ser feita em números ou letras. Essa inversão serve para poder multiplicar os números individuais por suas potências de 10, para montar o número final.

6. Funções Principais

6.1 lendoEntrada

Essa função é responsável por interpretar a entrada do usuário, e identificar se o usuário está usando comandos, como o de mostrar histórico ou ativar o alarme, ou entrando os numeros de temperatura e fumaça.

Como parâmetro de entrada, está o buffer, contendo a string digitada pelo usuário. São usados os registradores \$v0, \$v1, \$t0, \$t1, \$s0, para, respectivamente, guardar o valor da temperatura, guardar o valor da fumaça, contar as vezes lidas, guardar o caractere atual da string e guardar o topo da pilha auxiliar.

A pilha é usada para armazenar temporariamente os valores lidos de fumaça e temperatura. E o retorno é feito através dos registradores \$v0, \$v1, que retornam os valores de temperatura e fumaça.

A função analisa a entrada fornecida pelo usuário, verifica se há algum comando especial, e, caso negativo, interpreta a string como dois valores inteiros, que correspondem à fumaça e temperatura.

6.2 GravarLeitura

Essa função armazena a leitura no vetor de histórico correspondente, utilizando índice circular. Os parâmetros de entrada \$a0, \$a1 e \$a2 são respectivamente o valor de leitura, o total de leituras realizadas e o endereço base do vetor de destino.

Os registradores utilizados foram o \$a1 e \$a2, usados para o cálculo do índice e para guardar o endereço final de escrita.

A função é responsável por armazenar uma leitura no vetor de histórico. A fim de limitar o número de registros, é utilizado um índice circular calculado a partir do total de leituras. Dessa forma, as leituras superiores à décima leitura substituem as leituras mais antigas.

6.3 ListarUltimasLeituras

Essa função é responsável por listar na tela as últimas leituras de temperatura e fumaça armazenadas no histórico. Ela é acionada quando o usuário insere o comando 'l'.

Como parâmetro de entrada, a função utiliza o registrador \$t5, que armazena o total de leituras realizadas. São utilizados os registradores \$t0, \$t7, \$s1 e \$s2, sendo respectivamente

o índice do laço de repetição, um registrador auxiliar para cálculo de endereços, o endereço base do vetor de temperaturas e o endereço base do vetor de fumaça.

No caso de nenhuma leitura, a função exibe uma mensagem de erro informando que não há leituras disponíveis. Caso contrário, a função percorre os vetores e exibe os pares de valores correspondentes às leituras de temperatura e fumaça.

6.4 CalcularRisco

Essa função é responsável por calcular o nível de risco do sistema com base nos valores de temperatura e fumaça fornecidos.

Os parâmetros de entrada são os registradores \$a0 e \$a1, que representam respectivamente os valores de temperatura e fumaça. O registrador \$t1 é utilizado para armazenar valores intermediários durante o cálculo.

A função aplica pesos diferentes para a temperatura e fumaça, soma os resultados e normaliza o valor obtido. Caso o valor final ultrapasse o limite máximo permitido, o risco é limitado ao valor 100, garantindo que o sistema opere dentro da faixa estabelecida.

6.5 OutputAlarme

Essa função é responsável por exibir ao usuário o valor do risco calculado e o estado atual do sistema.

O parâmetro de entrada é o registrador \$a0, que contém o valor do risco. O registrador \$t1 é utilizado como cópia do risco para realizar comparações e definir o estado correspondente.

A partir do valor do risco, a função determina o estado do sistema, que pode ser normal, atenção, alerta ou evacuação. Cada estado é associado a uma faixa específica de risco, e a mensagem correspondente é exibida ao usuário.

6.6 adicionarAlarmeManualmente

Essa função é responsável por ativar manualmente o alarme em seu nível máximo, independentemente dos valores de temperatura e fumaça. O registrador \$t4 é utilizado para armazenar o valor fixo de risco igual a 100, representando uma situação crítica.

A função exibe na tela o risco máximo, o estado de evacuação e uma mensagem indicando que o alarme foi ativado manualmente. Essa função permite um acionamento manual do alarme, em casos de teste ou erro nos sensores.

6.7 Encerrar

Essa função é responsável por finalizar a execução do programa. Ela realiza uma chamada de sistema para encerrar o programa de forma adequada quando o usuário insere o comando 'e'.

7. Uso de IA

7.1 Ferramentas de IA utilizadas

ChatGPT.

7.2 Como a IA foi utilizada no projeto

As inteligências artificiais foram utilizadas para o aprendizado de recursos da linguagem assembly. Algumas funcionalidades do trabalho exigiram conhecimentos adicionais, como a implementação de vetores circulares, o uso de pilhas e de funções (sub-rotinas). Nesses casos, a IA foi utilizada para o ensino. Quanto ao uso para geração de código, foi acordado entre o grupo que as inteligências artificiais não são consistentes o suficiente para geração de código em assembly e estávamos confiantes em nossa habilidade de realizar o projeto definido.

8. Testes e Resultados

8.1 Casos de testes realizados

Foram utilizados diferentes casos de teste com o objetivo de validar o funcionamento do sistema em cenários variados. Inicialmente, foi considerado um caso de teste típico, no qual o usuário insere valores válidos de temperatura e fumaça no formato esperado pelo programa. Esse teste permitiu verificar a correta leitura dos dados, o cálculo do risco e a exibição do estado correspondente, além do armazenamento das leituras no histórico.

Também foram realizados testes extremos ou adversos, utilizando valores elevados de temperatura e fumaça, de modo que o cálculo inicial do risco ultrapassasse o limite máximo estabelecido. Observou-se que o sistema limita corretamente o valor do risco a 100, exibindo o estado de evacuação. Adicionalmente, foi testada a ativação manual do alarme por meio de comando específico, confirmando que o sistema responde adequadamente ao forçar o estado máximo de emergência.

Por fim, foram executados testes estruturais relacionados ao armazenamento das leituras. Nesse cenário, foram inseridas mais de dez leituras consecutivas para verificar o funcionamento do buffer circular, constatando-se que apenas as leituras mais recentes são mantidas. Também foi testada a solicitação de listagem sem registros prévios, resultando na exibição da mensagem de erro apropriada, o que confirma o correto tratamento de estados limite do sistema.

8.2 Screenshots e trechos de saída

Com o sistema implementado, foi possível realizar testes para simular o funcionamento do dispositivo embarcado. Observou-se que o programa realiza corretamente o cálculo da função de risco, apresentando como esperado os estados de severidade e os valores de risco. Isso pode ser observado na figura 1, que apresenta a entrada de valores que resultam em cada um dos estados como saída.

```
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
20 5
RISCO: 4/100
ESTADO: normal
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
50 20
RISCO: 12/100
ESTADO: atenção
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
100 100
RISCO: 41/100
ESTADO: alerta
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
150 200
RISCO: 75/100
ESTADO: evacuação
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
250 300
RISCO: 100/100
ESTADO: evacuação
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
```

Figura 2: Exemplos de entradas e saídas para cada estado de risco.

Note que, assim como o previsto, o valor do risco não ultrapassa 100, mesmo que o cálculo inicial resulte em valores maiores que 100, como no quinto caso.

Também foi feita a verificação da ativação manual de emergência pelo usuário. Com ela, observou-se como saída o estado de evacuação com valor de risco 100, como apresentado na figura 2.

```
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
b
RISCO: 100/100
ESTADO: evacuação
Alarme ativado manualmente
```

Figura 3: Exemplo de ativação manual de emergência.

Ademais, foram testadas duas funcionalidades essenciais. A primeira é de listagem das últimas leituras, possível a partir do uso de um buffer circular. O resultado é apresentado na figura 3. A segunda é a resposta à entradas inválidas, ou seja, informe de erro caso as entradas não sigam algum dos formatos especificados ou caso for solicitada a listagem das últimas leituras sem que haja registros prévios. A saída para esses casos é apresentada na figura 4.

```
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
l
(Temperatura | fumaça)
20 5
(Temperatura | fumaça)
50 20
(Temperatura | fumaça)
100 100
(Temperatura | fumaça)
150 200
(Temperatura | fumaça)
250 300
```

Figura 4: Listagem das últimas marcações feitas.

```

ERRO: O sistema detectou um erro na leitura de dados, verificar sensores.
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
a
ERRO: O sistema detectou um erro na leitura de dados, verificar sensores.
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
ab
ERRO: O sistema detectou um erro na leitura de dados, verificar sensores.
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
440
ERRO: O sistema detectou um erro na leitura de dados, verificar sensores.
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
440 200 300
ERRO: O sistema detectou um erro na leitura de dados, verificar sensores.
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
l
ERRO: Nenhuma leitura feita.
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.

```

Figura 5: Detecção de entradas inválidas.

Por fim, a Figura 5 ilustra o uso do comando “e” para encerrar o programa. Após sua execução, o sistema finaliza imediatamente a aplicação, impedindo a realização de novas entradas de dados pelo usuário.

```

ERRO: Nenhuma leitura feita.
Insira (Temperatura Fumaça) separadas por espaço OU
insira o código 'b' para ativação manual do alarme OU
o código 'l' para listar as ultimas leituras. Digite 'e' para encerrar.
e
-- program is finished running --

```

Figura 6: encerramento do programa com comando “e”.

8.3 Discussão dos resultados

A partir da análise dos resultados, conclui-se que o sistema se mostrou funcional, estabelecendo com sucesso a interação com o usuário, a partir de mensagens exibidas através das chamadas de sistema, com sentenças diretas e simples. Também, foram executadas com êxito as fórmulas de cálculo de risco e do estado de emergência, apresentando a saída na tela para o usuário, assim como a ativação manual. Ademais, foram feitas a validação da entrada, cálculos internos e exibição das mensagens, sem perda de dados e preservando os registradores por meio do uso adequado da pilha, o que permitiu a exibição das últimas leituras e a verificação de entradas válidas

9. Conclusão

O projeto em MIPS ajudou a colocar em prática conceitos importantes da disciplina de Arquitetura e Organização de Computadores. Com a implementação do programa, o grupo aprendeu como a pilha funciona, como um buffer circular pode ser implementado, como podemos transformar caracteres em números e como o código pode ser dividido em funções. O uso de vetores e o trabalho direto com endereços também ajudaram a compreender como os dados são manipulados em baixo nível.

No processo de desenvolvimento do código, desafios foram enfrentados pelo grupo devido às características da linguagem assembly. Um deles foi causado pela limitação de linguagens como assembly de exigir a definição do tipo das variáveis para receberem valores; isso demandou que os valores numéricos fossem tratados inicialmente como strings e depois fosse feita a conversão para o tipo inteiro. Além disso, houve problemas com o uso da variável \$ra, responsável por armazenar o endereço de retorno das chamadas. Seu valor precisou ser salvo numa variável auxiliar, pois, ao chamar uma função dentro de outra, uma nova instrução jal sobrescreve o conteúdo de \$ra, o que causava a perda do endereço de retorno original. Ao salvar esse valor temporariamente, garantiu-se que o fluxo de execução do programa fosse retomado corretamente após o término das sub-rotinas.

Com o desenvolvimento do projeto, foi possível perceber a contribuição da inteligência artificial no processo de aprendizagem de programação. A IA foi utilizada no esclarecimento de dúvidas de forma direta e pontual, tornando o aprendizado mais ágil e eficiente. Em especial, seu uso foi importante para compreender o funcionamento e a aplicação de vetores, pilhas, registradores e instruções específicas da arquitetura MIPS, contribuindo para um melhor entendimento dos conceitos envolvidos.

Assim, a simulação do monitoramento de temperatura e fumaça mostrou como sistemas embarcados fazem leituras de sensores, tratam informações e tomam decisões automáticas. Mesmo sendo um projeto didático, ele mostrou como a linguagem assembly pode ser usada para implementar programas com aplicações reais, neste caso, de monitoramento ambiental. Como extensão do projeto, é proposta a aplicação desta abordagem em sistemas funcionais e efetivos do mundo real com a associação de sensores externos e a um dispositivo que integre hardware e software. Além disso, mais funcionalidades podem ser implementadas ao sistema, como a instrução ao usuário de como proceder em cada caso e o informe das rotas de emergência.

10. Referências Bibliográficas

PATTERSON, David A.; HENNESSY, John L. *Organização e Projeto de Computadores: A Interface Hardware/Software*. 5. ed. Rio de Janeiro: Elsevier, 2014.

TAVARES, M. J.; BARBOSA, L. C. **Metodologias Ativas no Ensino de Hardware: Uma Revisão Sistemática**. *Revista Brasileira de Informática na Educação*, v. 27, n. 1, p. 112–125, 2019.

STALLINGS, W. **Arquitetura e Organização de Computadores**. 10. ed. São Paulo: Pearson, 2017.

WEBER, R. *Assembly MIPS: Arquitetura e Programação*. 2. ed. Porto Alegre: Sagra Luzzatto, 2008. (Caso não queira essa referência, posso remover — ela é adequada à parte de funções e pilha.)