EngIn Learn Framework (ELF)

Framework pedagógica genérica para unidades curriculares da Licenciatura em Engenharia Informática

Versão 1.0

"*Done with AI not done by AI*"
jenni

1. Fundamentos da *framework*

O ensino tem sido marcado por dificuldades persistentes, sobretudo no primeiro ano dos cursos de Engenharia Informática, onde os estudantes enfrentam a exigência de compreender simultaneamente conceitos teóricos abstratos e desenvolver competências práticas concretas. Muitos oscilam entre a memorização superficial da teoria e a execução mecânica de tarefas laboratoriais, sem conseguirem construir uma visão integrada do domínio. Esta constatação evidenciou a necessidade de conceber um modelo pedagógico inovador que assegurasse não apenas a aquisição de conhecimentos fundamentais, mas também o desenvolvimento do pensamento crítico, da autonomia e da capacidade de aplicação prática.

É neste contexto que surge o EngIn Learn Framework (ELF) (Figura 1)**,** um novo modelo pedagógico que combina aprendizagem ativa, avaliação formativa e integração criteriosa de inteligência artificial generativa no estudo teórico, preservando em simultâneo a autenticidade da prática laboratorial. A sua conceção assenta em três eixos fundamentais: a promoção da aprendizagem ativa, a estruturação progressiva do compromisso cognitivo através da Taxonomia de Bloom e do ICAP *framework*, e o uso responsável da IA generativa como mediadora de *feedback* imediato e de treino metacognitivo.

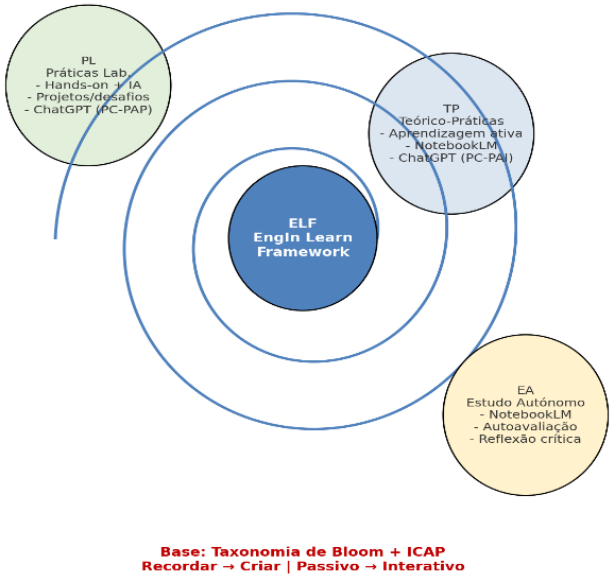

Figura 1 – EngIn Learn Framework (ELF).

A relevância da aprendizagem ativa na área das STEM encontra-se amplamente documentada. Meta-análises que sintetizam centenas de estudos demonstram que metodologias centradas no estudante, como a discussão estruturada, a resolução de problemas e o *peer instruction*, produzem melhorias significativas no desempenho académico e reduzem substancialmente as taxas de insucesso quando comparadas com aulas expositivas tradicionais. Freeman et al. (2014), num estudo de referência que abrangeu 225 comparações, concluíram que a aprendizagem ativa conduz a efeitos robustos e replicados em diferentes contextos. No domínio específico da engenharia, Prince (2004) e mais recentemente Deng et al. (2024) confirmam que estratégias como o *problem-based learning* e o *just-in-time teaching* estão associadas a ganhos consistentes em compreensão conceptual e transferência de conhecimentos. Do mesmo modo, Crouch e Mazur (2001), ao longo de dez anos de investigação sobre *peer instruction*, demonstraram aumentos sustentados na consolidação de conceitos básicos e na capacidade de resolução de problemas, evidências diretamente transponíveis para os objetivos de AOC, como a análise de desempenho, a compreensão de ISA e a exploração de *datapaths*.

Para além das metodologias ativas, o ELF encontra sustentação teórica no ICAP *framework* (Chi & Wylie, 2014), que distingue entre níveis de envolvimento passivo, ativo, construtivo e interativo, atribuindo a estes últimos um impacto mais profundo e duradouro na aprendizagem. Atividades que exigem que os estudantes expliquem raciocínios, comparem soluções ou gerem contraexemplos estão associadas a aprendizagens mais robustas do que aquelas em que o estudante se limita a seguir instruções. A Taxonomia de Bloom, na sua revisão por Krathwohl (2002), fornece a espinha dorsal curricular do ELF, ao articular os níveis cognitivos (Recordar,

Compreender, Aplicar, Analisar, Avaliar e Criar), com diferentes tipos de conhecimento: factual, conceptual, procedimental e metacognitivo. Esta estrutura permite o alinhamento claro entre objetivos, práticas pedagógicas e instrumentos de avaliação, favorecendo a progressão cognitiva dos estudantes de forma gradual e consistente.

Um terceiro pilar do modelo prende-se com a integração responsável da IA generativa. A literatura recente sublinha que, quando enquadrada de modo ético e pedagógico, a IA pode melhorar o desempenho académico e apoiar o desenvolvimento de disposições de ordem superior, sem efeitos negativos na motivação dos estudantes (Deng et al., 2024). Uma meta-revisão publicada no *International Journal of Educational Technology in Higher Education* recomenda precisamente que a adoção destas ferramentas seja feita com rigor, colaborativamente e com foco na responsabilidade pedagógica (Bond et al., 2024). O'Dea et al. (2024), numa revisão crítica, acrescentam que, embora a IA generativa ofereça oportunidades relevantes, como apoio à escrita e à explicação de conceitos, subsistem riscos de dependência que exigem um enquadramento refletido e orientado para o pensamento crítico. É essa orientação que o ELF adota ao integrar a IA no estudo teórico e ao utilizá-la de forma complementar no laboratório, sempre preservando a prática manual como espaço insubstituível para a consolidação de competências técnicas.

O modelo organiza-se em três dimensões interligadas. A componente teórica desenvolve-se nas aulas teórico-práticas, apoiada pelo Prompt Completo – Plano de Autoestudo Interativo (PC-PAI) no ChatGPT, que conduz o estudante num diálogo estruturado em formato pergunta–resposta, alternando questões de verdadeiro-falso, escolha múltipla, resposta curta e exercícios práticos, com *feedback* explicativo imediato e balanços quantitativos e qualitativos e sugestões de melhoria no estudo, por secção e no final da sessão. A prática laboratorial assenta em atividades *mão-na-massa*, onde a proficiência manual é crítica e não pode ser substituída, mas pode ser apoiada pelo Prompt Completo – Prompt de Apoio Prático (PC-PAP), que funciona como tutor complementar de raciocínio e comparação de soluções. Finalmente, o estudo autónomo assume um papel de consolidação e expansão, em que os estudantes recorrem às ferramentas de IA para revisão dirigida, autoavaliação progressiva e sugestões personalizadas de melhoria, promovendo a autonomia e a metacognição.

Na prática, o ELF funciona como um ciclo contínuo: a teoria fornece os fundamentos conceptuais, a prática laboratorial aplica e consolida esses conhecimentos, o estudo autónomo permite a sua revisão e expansão, e o processo retorna novamente à teoria, mas em níveis mais elevados de complexidade. Ao longo deste percurso, os estudantes percorrem de forma sistemática os diferentes níveis da Taxonomia de Bloom e são constantemente incentivados a manter um compromisso cognitivo profundo segundo o ICAP *framework*.

Em síntese, o ELF justifica-se por alinhar práticas ativas com evidência empírica robusta, estruturar o progresso cognitivo com base em Bloom e ICAP, e integrar a IA generativa de forma pedagógica e controlada, preservando a autenticidade e a exigência da prática manual em laboratório. O impacto esperado é a melhoria sustentada do desempenho, a redução das taxas de reprovação e o fortalecimento de competências críticas das unidades curriculares, contribuindo assim para uma formação mais sólida e completa dos futuros engenheiros informáticos.

2. Estrutura do ELF

Cada unidade curricular é organizada em três dimensões, que funcionam de forma cíclica e interligada:

Aulas Teórico-Práticas (TP)

- Objetivo: introdução e aprofundamento de conceitos fundamentais.
- Metodologias ativas: perguntas em tempo real, *think-pair-share*, miniprojectos.
- Ferramentas IA:
    - NotebookLM: organização dos conteúdos programáticos, mapas conceptuais dinâmicos, perguntas e repostas e resumos de áudio.
    - ChatGPT – Prompt Completo de Autoestudo (PC-PAI): simulação de tutoria interativa, com perguntas Verdadeiro-Falso, Escolha múltipla, Resposta curta e Problemas práticos.

Práticas Laboratoriais (PL)

- Objetivo: aplicação prática dos conceitos em ambientes experimentais.
- Metodologias ativas: *hands-on*, *problem-based learning*, pequenos desafios semanais.
- Ferramentas IA:
    - ChatGPT5 – Prompt de Apoio Prático (PC-PAP), adaptado à área (ex.: programação em C, Java, análise de algoritmos, entre outras).
    - Uso limitado e supervisionado: IA como apoio explicativo, nunca substituindo a execução manual/código pelo estudante.

Estudo Autónomo (EA)

- Objetivo: consolidação, revisão e metacognição.
- Metodologias ativas: autoavaliações, reflexões críticas, entre outras.
- Ferramentas IA:
    - NotebookLM: sumários, fichas personalizadas de revisão, resumos de aúdio.
    - ChatGPT – Autoavaliação e Feedback: simulação de testes, balanços quantitativos/qualitativos, recomendações de estudo.

3. Progressão Cognitiva (Bloom + ICAP)

O ELF integra a Taxonomia de Bloom como fio condutor:

- Recordar (definições, factos).
- Compreender (explicar em palavras próprias).
- Aplicar (resolver problemas).
- Analisar (comparar, decompor, justificar).
- Avaliar (crítica fundamentada de soluções).
- Criar (desenvolver projetos ou soluções originais).

Articulado com o ICAP *framework*, garante envolvimento ativo, construtivo e interativo, superando a aprendizagem passiva.

4. Ciclo de Aprendizagem ELF

1. Teoria → introdução de conceitos com IA como apoio explicativo e formativo.
2. Prática → aplicação direta em exercícios laboratoriais, com supervisão docente.
3. Estudo Autónomo → consolidação e *feedback* personalizado, em diálogo com a IA.
4. Iteração → o ciclo reinicia em níveis mais avançados, promovendo aprendizagem em espiral.

5. Avaliação formativa e sumativa

- Formativa: integrada nas sessões com IA (PC-PAI e PC-PAP), com balanços por nível Bloom.
- Sumativa: testes e projetos que avaliam desde Recordar →…→ Criar.
- Rubricas: baseadas em Bloom e ICAP, garantem transparência e progressão.

6. Benefícios do ELF

- Adaptabilidade a qualquer unidade curricular do curso.
- Integração natural de IA como copiloto de aprendizagem.
- Estrutura de progressão cognitiva clara e mensurável.
- Maior autonomia do estudante, redução da taxa de insucesso e desenvolvimento de competências críticas e criativas.

Referências

Freeman, S., et al. (2014). *Active learning increases student performance in science, engineering, and mathematics.* PNAS, 111(23), 8410–8415. doi:10.1073/pnas.1319030111.

Prince, M. (2004). *Does Active Learning Work? A Review of the Research.* Journal of Engineering Education, 93(3), 223–231. doi:10.1002/j.2168-9830.2004.tb00809.x.

Crouch, C. H., & Mazur, E. (2001). *Peer Instruction: Ten years of experience and results.* American Journal of Physics, 69(9), 970–977. doi:10.1119/1.1374249.

Chi, M. T. H., & Wylie, R. (2014). *The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes.* Educational Psychologist, 49(4), 219–243. doi:10.1080/00461520.2014.965823.

Krathwohl, D. R. (2002). *A Revision of Bloom's Taxonomy: An Overview.* Theory Into Practice, 41(4), 212–218. doi:10.1207/s15430421tip4104_2.

Deng, R., et al. (2024). *Does ChatGPT enhance student learning? A systematic review and meta-analysis of experimental studies.* Computers & Education, 105224. doi:10.1016/j.compedu.2024.105224.

Bond, M., et al. (2024). *A meta systematic review of Artificial Intelligence in Higher Education: a call for increased ethics, collaboration, and rigour.* International Journal of Educational Technology in Higher Education, 21, 9. doi:10.1186/s41239-023-00436-z.

O'Dea, X., et al. (2024). *Generative AI: is it a paradigm shift for higher education?* Studies in Higher Education (Special Issue).

BloomArch Learning Framework (BELF)

Versão 1.0

Arquitetura e Organização de Computadores

O BloomArch Learning Framework (BELF) constitui uma evolução aplicada e especializada do EngIn Learn Framework (ELF), descrito anteriormente. Enquanto o ELF foi concebido como uma *framework* pedagógica genérica para apoiar as unidades curriculares da Licenciatura em Engenharia Informática, integrando metodologias ativas, progressão cognitiva estruturada (Bloom + ICAP) e o uso responsável da inteligência artificial generativa, o BELF apropria-se destes fundamentos e adapta-os especificamente ao domínio da Arquitetura e Organização de Computadores (AOC).

Assim, o BELF é resultado do ELF por duas razões centrais:

1. Herança de princípios estruturantes: ambos os modelos assentam na aprendizagem ativa, na metacognição e no desenvolvimento do pensamento crítico, articulando de forma explícita a Taxonomia de Bloom com ferramentas de IA (ChatGPT e NotebookLM) como suporte pedagógico.
2. Especialização curricular: enquanto o ELF define uma lógica geral de progressão teórica, prática e autónoma, o BELF operacionaliza esta lógica em atividades, rotinas e *prompts* ajustados ao ensino e aprendizagem da AOC, explorando, por exemplo, exercícios de Assembly com tutoria detalhada via IA e mapas conceptuais orientados para os conteúdos específicos da disciplina.

Deste modo, o BELF pode ser entendido como uma instância concreta e aplicada do ELF, em que a estrutura genérica do primeiro é transposta para um contexto disciplinar, demonstrando como o modelo pode ser escalado e adaptado a diferentes áreas, mantendo os princípios nucleares do *framework* original, mas ajustando estratégias, ferramentas e rotinas às necessidades específicas da unidade curricular de AOC.

1. Princípios orientadores

- **Aprender fazendo**: *peer instruction*, *think–pair–share*, resolução de problemas, laboratórios.
- **Bloom**: *Recordar* → *Compreender* → *Aplicar* → *Analisar* → *Avaliar* → *Criar*, revisitados em cada tópico.
- **Metacognição e pensamento crítico**: ciclos curtos de "planeamento → execução → reflexão", comparação de soluções e critérios de qualidade explícitos.
- **IA como tutor e laboratório**: ChatGPT para diálogo socrático e geração de *feedback* para a teoria e prática; NotebookLM para estudo dirigido, Relatórios (Guias de Estudo), Resumo de áudio e Mapa mental.

2. Integração tecnológica (NotebookLM + ChatGPT)

2.1 NotebookLM

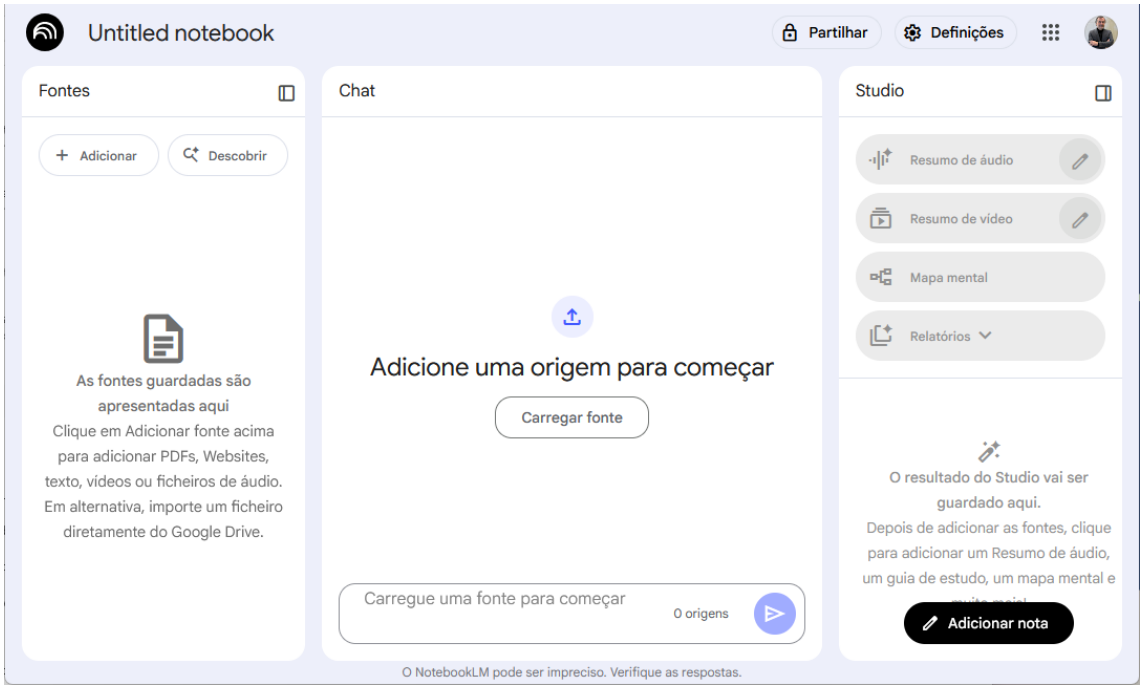

Figura 2 - NotebookLM

Usar todas as funcionalidades do NotebookLM (Figura 2) de forma intencional:

1. **Fontes**: carregar diapositivos, capítulos de livros, *links* do Youtube referentes aos conteúdos programáticos.
2. **Resumos**: gera mapas (mentais) de tópicos por secção.
3. **Resumo de áudio**: criar um resumo em "podcast" para revisão auditiva antes da aula.
4. **Perguntas e Respostas**: interrogar o conteúdo carregado com referência automática às páginas/fontes.
5. **Relatórios (Guias de estudo)**: Texto resumo com perguntas com tipologias diferentes e respostas.
6. **Relatórios (Linha cronológica)**: organizar cronologias ou tabelas.
7. **Notas**: fixar exemplos típicos para estudo.

Sugestão de rotina pré-aula (15–25 min):

- Abrir NotebookLM → *Relatórios (Guias de estudo)* → *Resumo de áudio* (5–7 min) → 2 Perguntas e Respostas dirigidas ao conteúdo carregado.

2.2 ChatGPT

Usar o *prompt* PC-PAI completo da Figura 3, para estudo dos conteúdos teóricos em modo diálogo pergunta–resposta, com *feedback imediato* e dificuldade crescente:

Prompt Completo – Plano de Autoestudo Interativo (PC-PAI)

Contexto e papel
Atua como um especialista sénior na área do conhecimento abordado nos ficheiros carregados e também especialista em desenvolvimento de planos pedagógicos de autoaprendizagem.
A tua função é criar um plano de autoestudo interativo, baseado no(s) ficheiro(s) fornecido(s), conduzido em formato de diálogo pergunta–resposta com o aprendiz.

Estrutura do processo
1. Análise inicial:
• Lê os ficheiros carregados.
• Divide os conteúdos em secções/tópicos principais.
• Apresenta ao aprendiz a lista das secções disponíveis e pergunta: "Por qual secção gostarias de começar?"
2. Condução da aprendizagem dentro de uma secção
• Define objetivos de aprendizagem.
• Conduz o estudo pergunta a pergunta, num Ciclo interativo:
(1) coloca pergunta
(2) aguarda resposta
(3) dá feedback (certo/errado, explicação, exemplo prático)
(4) Pergunta se o aprendiz quer mais perguntas deste tópico (com maior complexidade) ou se prefere avançar para a frente dentro da secção.
• Tipos de perguntas usados em rotação:
- Verdadeiro/Falso
- Escolha Múltipla (4 opções)
- Resposta curta/desenvolvimento
- Exercícios práticos (problemas, cálculos, casos reais).
• Níveis de dificuldade:
- Básico → Intermédio → Avançado.
3. Encerramento da secção
Quando a secção for concluída (ou o aprendiz indicar que quer parar):
• Apresenta um balanço quantitativo (respostas certas/erradas).
• Faz uma avaliação qualitativa (iniciante, intermédio, avançado).
• Dá sugestões de melhoria personalizadas.
• Pergunta ao aprendiz o que deseja fazer a seguir: "Queres continuar nesta secção até te sentires satisfeito, escolher outra secção para explorar, ou terminar o estudo?"
4. Encerramento global do plano de estudo
Se for escolhido terminar o estudo:

• Apresenta um resumo global do desempenho, incluindo pontos fortes e pontos a melhorar.
• Dá uma avaliação qualitativa final (nível geral do aprendiz).
• Sugere um plano de estudo complementar (leituras, exercícios, recursos).

Regras fundamentais:
• Sempre um diálogo interativo: uma pergunta → resposta → feedback.
• Explicações detalhadas e pedagógicas em todas as respostas.
• Complexidade crescente dentro de cada secção.
• Avaliação quantitativa + qualitativa em cada secção e no final global.
• O aprendiz tem sempre a liberdade de escolha:
o permanecer na mesma secção,
o avançar para outra,
o ou terminar o estudo.

Figura 3 – Prompt Completo – Plano de Autoestudo Interativo (PC-PAI).

Rotina pós-aula (20–30 min):

- Repetir 1–2 secções no ChatGPT com o *prompt* PC-PAI pedindo explicitamente perguntas que provoquem análise e avaliação (ex.: "compara duas soluções, identifica assunções, deteta falácias de raciocínio").

Usar o *prompt* PC-PAA completo da Figura 4 para estudo dos conteúdos práticos, nomeadamente a programação em Assembly, escrevendo, explicando e corrigindo programas passo a passo.

Prompt Completo – Plano de Autoestudo Assembly (PC-PAA)

Pedir o enunciado entre **[ ]**, a tua solução entre **{ }**, gerar uma resolução

#modelo e comparar, com feedback pedagógico e correções.

QUERO QUE ATUES COMO UM/UMA TUTOR(A) DE MIPS (ASSEMBLY): AVALIAÇÃO, RESOLUÇÃO E FEEDBACK PEDAGÓGICO

OBJETIVO
Ajudar-me a aprender MIPS escrevendo, explicando e corrigindo programas passo a passo. Vamos trabalhar sempre com convenções típicas de MARS (syscalls padrão). Usa português de Portugal.

COMO INTERAGIMOS (FORMATO OBRIGATÓRIO)
1) Eu envio o ENUNCIADO do problema entre parênteses retos: [...].
2) Eu submeto a MINHA SOLUÇÃO MIPS (ou pseudocódigo, se ainda não tiver MIPS) entre chavetas: { ... }.
3) Tu preparas:
A) Análise do enunciado.
B) Resolução modelo (com código MIPS comentado).

C) Execução mental (dry-run) com casos de teste.
D) Comparação detalhada entre a tua solução e a minha { }.
E) Lista de erros/risco e COMO corrigir, com explicações pedagógicas.
F) Sugestões de melhoria e exercícios de consolidação.

REGRAS DE APRESENTAÇÃO
- Sê rigoroso, mas pedagógico. Explica o "porquê" de cada correção.
- Quando mostrares código, usa bloco ```mips com comentários linha a linha.
- Assume MARS. Se usares syscalls, inclui a tabela de registos usados (v0, a0, etc.).
- Evita pseudoinstruções quando relevante; se as usares, explica a expansão (ex.: move → add $t0,$zero,$t1).
- Durante a comparação, cita pequenos trechos do meu código (máx. 2–3 linhas por citação) e mostra a alternativa correta.

CONTEÚDO DETALHADO QUE DEVES PRODUZIR

[A] ANÁLISE DO ENUNCIADO
- Reescreve com as tuas palavras o objetivo do problema.
- Identifica: entradas, saídas, restrições (tamanho, limites numéricos, terminação), casos de fronteira e erros comuns.
- Escolhas de desenho: estrutura de dados, uso de .data/.text, labels, necessidade (ou não) de pilha e chamadas.

[B] RESOLUÇÃO MODELO (PASSO A PASSO)
1. Ideia algorítmica em alto nível (em 5–10 linhas).
2. Mapa de registos que vais usar (v0/v1, a0–a3, t0–t9, s0–s7, sp, ra), indicando persistência (salvaguarda de $s* na pilha, se aplicável).
3. Esqueleto do programa:
  - .data (constantes, buffers, strings) com comentários.
  - .text (main): inicialização, ciclo/condições, I/O (syscalls), finalização limpa (li $v0,10; syscall).
4. Código MIPS comentado, com foco em:
  - Loads/stores corretos (lb/lh/lw vs sb/sh/sw), alinhamento e tipos.
  - Comparações e branches (beq/bne/bgtz/blez/slti, etc.), saltos limpos com labels claros.
  - Uso correto de syscalls (tabela abaixo).
  - Se houver função auxiliar: prólogo/epílogo (ajuste de $sp, save/restore de $ra e $s*), passagem de argumentos ($a0–$a3) e retorno ($v0).
5. Tabela rápida de syscalls (MARS):
  - imprimir inteiro: $v0=1, $a0=valor
  - imprimir string: $v0=4, $a0=endereço
  - ler inteiro: $v0=5 → resultado em $v0
  - ler string: $v0=8, $a0=buffer, $a1=tamanho
  - terminar: $v0=10
6. Dry-run (execução mental) com 2–3 casos:
  - Inclui pelo menos: caso normal, caso limite (fronteira), caso "patológico" (ex.: entrada zero ou negativa).
  - Mostra a evolução dos registos relevantes e memória/buffer, em bullets.

[C] COMPARAÇÃO COM A MINHA SOLUÇÃO { } (ANÁLISE DETALHADA)
- Sumário rápido: em 3–5 bullets, o que está certo e o que está errado no meu código.

- Quadro de divergências (tabela textual):
  * Aspeto (segmentos, labels, I/O, aritmética, branches, pilha, chamadas)
  * O que eu escrevi (extrato curto entre aspas)
  * Problema observado (preciso e objetivo)
  * Correção exata (linha(s) MIPS corrigida(s) ou alternativa)
  * Justificação pedagógica (porquê funciona; referência ao ISA ou convenção)
- Verificações específicas:
  1) .data/.text corretos? Labels únicos e significativos?
  2) Tipos e tamanhos: match entre lw/sw vs lb/sb; endereços corretos?
  3) Condições e saltos: ramos alcançáveis? Labels de saída bem colocados?
  4) Registos: $t* livres; $s* preservados; $a*/$v* usados corretamente?
  5) Pilha: $sp ajustado e restaurado; alinhamento; save/restore de $ra/$s*.
  6) Syscalls: valores em $v0/$a0… corretos; strings terminadas com 0x00 (se necessário).
  7) Estilo: comentários úteis, nomes de labels claros, ausência de "gotos" confusos.
- Para cada erro, dá UMA correção mínima e UMA alternativa opcional (se existir), explicando vantagens/desvantagens.

[D] QUALIDADE E DESEMPENHO
- Complexidade aproximada (laços/iterações).
- Notas de eficiência: redução de loads/stores, reuso de registos, evitar branches desnecessários (branch delay slots não simulados em MARS, mas comenta a ideia).
- Robustez de I/O e validação simples (se o enunciado o exigir).

[E] SUGESTÕES DE MELHORIA (EM CAMADAS)
- Nível 1 (essenciais): correções mínimas para fazer passar os testes.
- Nível 2 (boas práticas): melhores nomes, estrutura modular, comentários padrão.
- Nível 3 (avançado): funções separadas com prólogo/epílogo, testes adicionais, manipulação de arrays/strings mais robusta.

[F] EXERCÍCIOS DE CONSOLIDAÇÃO (3 itens)
- Variantes do problema que reforcem: 1) aritmética/branches, 2) buffers/strings, 3) stack/calls.

ESTILO DE SAÍDA
- Usa secções com títulos [A]…[F].
- Código em ```mips (sem outras linguagens).
- Explicações curtas por linha de código (comentários).
- Onde for útil, mostra equivalência de pseudoinstruções → instruções reais.

PRONTO/A PARA COMEÇAR
1) Envia o ENUNCIADO entre [ ].
2) Depois, envia a tua SOLUÇÃO entre { }.
3) Eu devolvo a análise, a resolução modelo, a comparação detalhada e as correções pedagógicas.

Figura 4 – Prompt Completo – Plano de Autoestudo Assembly (PC-PAA).

Rotina durante a aula:

Utilizar no ChatGPT com o *prompt* PC-PAA pedindo explicitamente através da introdução desenvolvido pelo estudante e da solução desenvolvida pelo ChatGPT para

provocar uma análise e avaliação do código produzido (ex.: “compara as duas soluções e sugerir a correção de erros ou sugerir resoluções alternativas”).

3. Mapa curricular com objetivos e atividades ativas + IA

Estrutura por tópico: Objetivos + Atividades ativas + Utilização da tecnologia (NotebookLM e/ou ChatGPT)

1. **Introdução à AOC**
    - **Bloom**: recordar a distinção arquitetura vs. organização; compreender níveis de abstração; aplicar mapeando componentes a funções.
    - **Ativas**: *Think–pair–share* sobre “O que é performance?”; Mapa mental colaborativo usando o NotebookLM.
    - **NotebookLM**: Relatórios (Guia de estudo) “arquitetura vs. organização”; *quiz* de termos.
    - **ChatGPT**: diálogo socrático com o *prompt* PC-PAI.
2. **Representação de informação**
    - **Bloom**: converter entre bases; analisar *overflow*; avaliar erros de arredondamento.
    - **Ativas**: práticas de conversão (*bin/hex/2’s comp/IEEE-754*)”.
    - **NotebookLM**: Perguntas e Respostas de passos de conversão.
    - **ChatGPT**: problemas graduais (*overflow*, *edge cases* de IEEE-754).
3. **Circuitos lógicos combinacionais**
    - **Bloom**: derivar *truth tables*; simplificar (álgebra/K-maps); criar circuitos (codificador/decodificador/ALU simples).
    - **Ativas**: revisão por pares de simplificações.
    - **NotebookLM**: coleções de exemplos canónicos com passos; Perguntas e Respostas de leis.
    - **ChatGPT:** gerar e corrigir *minterms/maxterms*, pedir explicação alternativa.
4. **Organização dos Computadores**
    - **Bloom**: identificar caminhos de dados e controlo; analisar impactos de escolhas (barramentos, sinais).
    - **Ativas**: revisão por pares sobre conceitos.
    - **NotebookLM**: esquemas anotados; Relatórios (Linha do tempo) evolução de microarquiteturas.
    - **ChatGPT**: questões “E se…?” sobre variantes de controlo.
5. **Noções básicas de desempenho de processadores**
    - **Bloom**: aplicar fórmulas (CPI, *speedup*); analisar *bottlenecks*; avaliar compensações (*pipeline* vs. frequências).
    - **Ativas**: *peer instruction* com *clickers*; mini-estudos de caso.

- **NotebookLM**: Resumo de áudio; Relatórios (Guisa de estudo); exercícios com soluções.
- **ChatGPT**: cálculo passo a passo e verificação de raciocínio.

6. **Conjunto de instruções (ISA)**
   - **Bloom**: classificar tipos de instrução e endereçamento; traçar execução; comparar CISC vs. RISC.
   - **Ativas**: "traduz da ALU para a ISA"; *code tracing* em pseudo-assembly.
   - **NotebookLM**: Resumo de áudio sobre ISA.
   - **ChatGPT**: desafios de *tracing* e previsão de estados de registos/memória.
7. **Organização de uma unidade de processamento**
   - **Bloom**: mapear micro-operações; analisar *datapath*; criar microprogramas simples.
   - **Ativas**: revisão por pares sobre conceitos sobre registos (MAR/MDR/PC/IR).
   - **NotebookLM**: Perguntas e Respostas; Relatórios (Guias de estudo).
   - **ChatGPT**: perguntas guiadas com "porquê" e contraexemplos.
8. **Sistemas de memória**
   - **Bloom**: calcular AMAT; analisar localidade; avaliar políticas (*write-through/back*, mapeamentos).
   - **Ativas**: simulação de acessos; debate "*cache* inclusiva vs. exclusiva".
   - **NotebookLM**: Resumo de áudio.
   - **ChatGPT**: problemas numéricos e *what-if*.

4. Modelo de aula/semana

- **Pré-aula (NotebookLM, 20 min)**: *Audio Overview* + *Relatórios (Guias de estudo)*.
- **Aula (60–120 min)**:
  1. *Peer instruction* (Verdadeiro-Falso/Escolha Múltipla) → discussão.
  2. Mini-laboratório (Em função da temática).
  3. *Think–pair–share*: análise de um erro comum.
- **Pós-aula (ChatGPT, 20–30 min)**: nas aulas TP usar o *prompt PC-PAI* para 1 secção → pedir aumento de dificuldade → obter *feedback* e plano de melhoria. Nas aulas PL, na aprendizagem de Assembly usar o *prompt PC-PAA* para corrigir exercícios complementares aos da aula.
- **Meta reflexão (5 min)**: "O que aprendi? Onde errei? Que suposições usei?"

5. Avaliação e acompanhamento

- **Formativa**: Relatórios (Guias de estudo) em NotebookLM, diálogo no ChatGPT (regista dificuldades recorrentes).
- **Sumativa**: Dois testes e um trabalho prático de grupo.

- **Rubricas de pensamento crítico** (síntese, identificação de pressupostos, comparação de soluções, justificações com evidência).
- **Ética na IA**: usar IA para compreender e não para copiar; citar quando relevante; validar resultados.

6. Como promover pensamento crítico com IA

- **Perguntas de "porquê/como sabes?"** no ChatGPT; exige justificar com evidências das fontes do NotebookLM.
- **Comparar duas soluções**: pedir ao ChatGPT que avalie *trade-offs* (nº de portas vs. atraso, performance vs. energia).
- **Refutar a resposta da IA**: obriga o estudante a encontrar contraexemplos nas fontes do NotebookLM.
- **"*Explain-Back*"**: o estudante explica a solução de volta, o ChatGPT verifica lacunas e gera *checklist* de correção.

7. Plano de estudo complementar (sugestão)

- **Antes de cada aula**: Resumo de áudio + Relatórios (Guias de estudo) (NotebookLM).
- **Depois de cada aula**: 12–18 min de diálogo com o *prompt PC-PAI no* ChatGPT, focando erros cometidos para os conteúdos programáticos das aulas TP e 12–18 min de diálogo com o *prompt PC-PAA no* ChatGPT, para melhor compreender os erros de programação.
- **Semanal**: 1 *quiz* de 10 questões no Perguntas e Respostas (NotebookLM) + 1 resolução de problemas complementares.
- **Revisão**: sessões *interleaved* (misturar tópicos), para reforçar retenção e transferência.

BloomArch Learning Framework

2ª semana
Versão 1.0

Conteúdo

- **Aula TP (2h)** → *Introdução à Arquitetura e Organização de Computadores*
- **Aula Laboratorial (2h)** → *Representação da Informação*
- **Plano pedagógico detalhado** (2h TP + 2h Lab)
- **Folhas de trabalho** (com soluções)
- **20 questões por tópico** (para usar no NotebookLM/ChatGPT)
- **20 exercícios laboratoriais resolvidos** (*Representação da Informação*, soluções passo a passo).

1) Aula Teórico-Prática (2h)

Tema: Introdução à Arquitetura e Organização de Computadores

Objetivos (Bloom)

- **Recordar**: diferenças entre arquitetura e organização; níveis de abstração.
- **Compreender**: papel do ISA; relação hardware/software.
- **Aplicar**: exemplos de *trade-offs* (latência vs. *throughput*).
- **Analisar**: diferentes definições de desempenho.
- **Avaliar**: impacto de decisões arquitetónicas.

Estrutura da Aula (2h)

- **10 min** — *Icebreaker*: **Perguntas e Respostas** de Verdadeiro-Falso (NotebookLM).
- **30 min** — Exposição dialogada com exemplos.
- **60 min** — *Peer instruction* com 6 questões Múltipla Escolha (NotebookLM).
- **20 min** — Síntese interativa (ChatGPT) → 3 questões Verdadeiro-Falso e/ou Múltipla Escolha + 1 exercício prático.

2) Aula Prática Laboratorial (2h)

Tema: Representação da Informação

Objetivos (Bloom)

- **Recordar**: sistemas de numeração (binário, decimal, hexadecimal).
- **Compreender**: complemento para 1 e 2; intervalos de representação.
- **Aplicar**: conversões entre bases.
- **Analisar**: casos de *overflow* e *underflow*.
- **Avaliar**: impacto da precisão (inteiros vs. ponto flutuante).
- **Criar**: algoritmos para converter números e verificar *overflows*.

Estrutura da Aula (2h)

- **15 min** — Introdução prática: revisão rápida de sistemas de numeração.

- **90 min** — Exercícios dirigidos (folha de trabalho + correção no quadro).
- **15 min** — Reflexão + resumo (usar NotebookLM (Perguntas e Respostas) + ChatGPT para explicações adicionais).

Folhas de Trabalho

Folha de Trabalho 1 (TP) — Introdução à AOC

Parte A — Questões de compreensão

1. Defina Arquitetura e Organização (exemplo real de diferença).
2. Quais os 3 níveis de abstração principais na computação?
3. Explicar com exemplo: *latência vs throughput*.

Parte B — Aplicação prática

4. Se um processador A executa 1B de instruções em 2s e outro B em 3s, qual o mais rápido?
5. Um CPU tem frequência de 3 GHz e CPI médio de 1,5. Qual o *throughput* (instr/s)?

Folha de Trabalho 2 (Lab) — Representação da Informação

Parte A — Conversões

1. Converter 45_{10} em binário e hexadecimal.
2. Converter 101101_2 para decimal e hexadecimal.

Parte B — Complemento e *overflow*

3. Representar −7 em 2's *complement* com 8 bits.
4. Calcular $100_{10} + (-50_{10})$ em complemento para dois (8 bits).

Parte C — Ponto flutuante

5. Representar $5{,}75_{10}$ em IEEE-754 precisão simples (passos).
6. Indicar erro de arredondamento ao representar $0{,}1_{10}$ em binário.

Questões complementares para importar no NotebookLM/ChatGPT

Questões – Introdução à AOC

1. Verdadeiro-Falso: Arquitetura refere-se a detalhes de implementação física.
2. Verdadeiro-Falso: Organização = conjunto de instruções.
3. Identificar dois exemplos de *trade-off* arquitetónico.
4. Definir ISA.
5. Qual a diferença entre arquitetura CISC e RISC?
6. Verdadeiro-Falso: O desempenho é medido apenas pela frequência do relógio.
7. Calcular o *throughput* de CPU a 2 GHz, CPI=1,2.
8. O que significa "lei de Amdahl"?
9. Qual destes é fator de desempenho?
    a) Frequência
    b) Energia
    c) Cores
    d) Todas
10. Explique "*pipeline*".

11. Verdadeiro-Falso: *Pipeline* aumenta latência, mas reduz *throughput*.
12. Dê um exemplo de paralelismo a nível de instrução.
13. Explique diferença entre escalabilidade vertical e horizontal.
14. Verdadeiro-Falso: Um processador com CPI=2 é sempre pior que CPI=1.
15. Explique a diferença entre tempo de resposta e débito.
16. Dar exemplo real onde *throughput* importa mais que latência.
17. Verdadeiro-Falso: Mais núcleos = sempre mais rápido.
18. Explique a relação desempenho/eficiência energética.
19. Defina *benchmark*.
20. Explicar: "GHz não é tudo".

Questões – Representação da Informação

1. Converter 25_{10} para binário.
2. Converter 1101_2 para decimal.
3. Converter $3A_{16}$ para decimal.
4. Converter 101101_2 para hex.
5. Representar −5 em 8 bits (2's *complement*).
6. Verdadeiro-Falso: Em 8 bits, intervalo de 2's *complement* é −128 a +127.
7. Verdadeiro-Falso: Em complemento para 1, existe representação duplicada do zero.
8. Converter 255_{10} para binário de 8 bits.
9. Qual o resultado: $11111111_2 + 00000001_2$ em 8 bits (2's *comp*)?
10. Representar $0{,}5_{10}$ em IEEE-754.
11. Representar $7{,}25_{10}$ em IEEE-754.
12. O que significa *overflow* em complemento para 2?
13. Converter -128_{10} em 8 bits.
14. Qual a diferença entre sinal-magnitude e complemento para 2?
15. Verdadeiro-Falso: O zero tem representação única em complemento para 2.
16. Converter $0{,}1_{10}$ em binário (4 casas).
17. Qual o valor decimal de 11111111_2 (2's *comp*, 8 bits)?
18. Intervalo 16 bits (2's *comp*) é:
    A) −255 a +255
    B) −32768 a +32767
    C) 0 a 65535
    D) Nenhuma
19. Explicar por que não se pode representar exatamente 0,1 em binário.
20. Dar exemplo de *underflow* em ponto flutuante.

Exercícios de Laboratório — Representação da Informação

Conversões básicas

1. 42_{10} → binário
2. 101101_2 → decimal
3. $7A_{16}$ → decimal
4. 255_{10} → hex
5. 110010_2 → hex

Complemento para dois

6. −6 (8 bits)
7. Soma: 5 + (−3) (4 bits)
8. *Overflow*: 100 + 50 (8 bits)
9. Valor de 11111111_2 (8 bits, 2's comp)
10. Valor de 10000000_2 (8 bits, 2's comp)

Casos de o*verflow*

11. 01111111_2 (127) + 00000001_2 (1)
12. 10000000_2 (−128) − 1

Ponto flutuante IEEE-754

13. $1{,}5_{10}$ → binário
14. $2{,}25_{10}$ → binário
15. $0{,}5_{10}$ → binário
16. $7{,}25_{10}$ → binário

Aplicados

17. Converter $0{,}1_{10}$ em binário (8 casas)
18. Arredondamento de 0,1 → causa erro acumulado.
19. Qual o intervalo representável em 16 bits (2's comp)?
20. Dar um exemplo real de *underflow*

BloomArch Learning Framework

3ª semana
Versão 1.0

Conteúdo

- **Aula TP (2h)** → *Noções básicas de circuitos lógicos*
- **Aula Laboratorial (2h)** → *Noções básicas de circuitos lógicos*
- **Plano pedagógico detalhado** (2h TP + 2h Lab)
- **Folhas de trabalho** (com soluções)
- **20 questões por tópico** (para usar no NotebookLM/ChatGPT)
- **20 exercícios laboratoriais resolvidos** (*Representação da Informação*, soluções passo a passo).

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- Compreender os conceitos de **portas lógicas** (AND, OR, NOT, NAND, NOR, XOR, XNOR).
- Analisar **tabelas de verdade** e reconhecer equivalências lógicas.
- Aplicar **leis da álgebra de Boole** para simplificar expressões.
- Relacionar a teoria com implementações práticas em hardware digital.

Estratégia pedagógica (BALF)

- **Aprendizagem ativa**: explicação intercalada com pequenos problemas em sala.
- **Uso do PC-PAI** (Prompt Completo – Plano de Autoestudo Interativo) no ChatGPT → questionário interativo.
- ***Feedback* imediato** em regime de diálogo.
- **Progressão cognitiva** segundo Bloom: recordar → aplicar → analisar.

2. Folha de Trabalho

Parte A – Verdadeiro/Falso

1. A porta NAND é a negação da porta AND.
2. Uma porta XOR é verdadeira quando as entradas são iguais.
3. A expressão A+A·B simplifica para A.
4. O complemento de uma variável é representado por um apóstrofo (').
5. A álgebra de Boole só é usada em eletrónica.

Parte B – Escolha múltipla

6. Qual a tabela de verdade correta da porta XOR?
    - A) 00→0, 01→0, 10→0, 11→1
    - B) 00→0, 01→1, 10→1, 11→0
    - C) 00→1, 01→0, 10→0, 11→1
    - D) 00→1, 01→1, 10→1, 11→0
7. Qual expressão é equivalente a (A+B)'?
    - A) A'+B'

- B) A’·B’
- C) AB
- D) (AB)’

8. Qual das seguintes é uma identidade da álgebra de Boole?
   - A) A+0=A
   - B) A+1=A
   - C) A·0=A
   - D) A·1=0
9. Se A=1, B=0, qual o valor de A·B+A’?
   - A) 1
   - B) 0
   - C) A
   - D) B
10. O circuito OR com duas entradas tem quantas combinações possíveis?
    - A) 2
    - B) 3
    - C) 4
    - D) 8

Parte C – Resposta curta

11. Explique a diferença entre XOR e XNOR.
12. Simplifique A·(A+B).
13. Construa a tabela de verdade para (A·B)’.
14. Porque é importante a álgebra de Boole na computação digital?
15. Simplifique A·B+A·B’.

Parte D – Exercícios práticos

16. Construa a tabela de verdade de F=A+BC.
17. Simplifique F=(A+B)(A+C).
18. Projete o circuito de F=AB+A’C.
19. Dê um exemplo prático do uso da porta XOR.
20. Mostre como implementar A+B apenas com NANDs.

3. Questões Complementares (NotebookLM/ChatGPT)

Verdadeiro/Falso

1. A porta NOR é a negação da porta NAND.
2. O valor de A+A’ é sempre 1.
3. A expressão AB+A’B simplifica para B.
4. A porta XNOR dá 0 quando as entradas são iguais.
5. Uma tabela de verdade com 3 variáveis tem 8 linhas.

Escolha múltipla

1. O resultado de A+1 é:
   - A) A
   - B) 0
   - C) 1
   - D) A'

2. O resultado de A·0 é:
    - A) 0
    - B) A
    - C) 1
    - D) A'
3. Qual destas é absorção?
    - A) A+AB=A
    - B) A·A=A²
    - C) A+A=A²
    - D) A+0=0
4. Qual expressão representa XOR em termos de OR, AND e NOT?
    - A) (A+B)(AB)
    - B) (A'B + AB')
    - C) (A+B)'
    - D) (AB)'
5. Qual a expressão equivalente a (AB)'?
    - A) A'+B'
    - B) A'+B
    - C) A+B'
    - D) AB

Resposta curta

11. Construa a tabela de verdade para A+B'.
12. Explique porque NAND e NOR são chamadas portas universais.
13. Simplifique AB+BC+AC.
14. Qual a diferença prática entre usar NOR e usar NAND para construir circuitos?
15. Explique como A+A'=1 representa um princípio fundamental da lógica.

Exercícios práticos

16. Construa a tabela de F=(A+B)(A'+B).
17. Simplifique (A+B)(A'+B).
18. Implemente um circuito para F=ABC+A'B'.
19. Mostre como realizar NOT usando apenas NAND.
20. Projete um circuito que deteta quando duas entradas são iguais (XNOR).

BloomArch Learning Framework

4ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**
- **20 questões por tópico (40 no total)**
- **20 exercícios laboratoriais com soluções detalhadas**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- **Recordar**: portas lógicas (NOT, AND, OR, NAND, NOR, XOR, XNOR), formas canónicas (SOP, POS), minterms/maxterms, Leis de De Morgan.
- **Aplicar**: construir tabelas-verdade; obter expressões canónicas; simplificar por álgebra e mapas de Karnaugh (3–4 variáveis).
- **Analisar**: comparar soluções (nº de portas/níveis), escolher implementação mais eficiente; justificar agrupamentos em K-maps.

Estratégia pedagógica (BALF)

- **Aprendizagem ativa**: explicação intercalada com pequenos problemas em sala.
- **Uso do PC-PAI** (Prompt Completo – Plano de Autoestudo Interativo) no ChatGPT → questionário interativo.
- ***Feedback* imediato** em regime de diálogo.
- **Progressão cognitiva** segundo Bloom: recordar → aplicar → analisar.

Folha de trabalho

Parte A — Recordar

1. Definir SOP e POS com um exemplo simples.
2. Escrever as Leis de De Morgan.
3. Explicar a diferença entre *minterm* e *maxterm*.

Parte B — Aplicar

4. A partir da tabela-verdade F(A,B) com F = 1 para m = {1,2}, escrever a SOP canónica.
5. Simplificar F(A,B,C) =A.B´ + A'.B + A.B.C' (álgebra).
6. Obter (F) mínima por K-map (3 variáveis) para m={1,3,5,7}.

Parte C — Analisar

7. Compare duas implementações de (XOR):
    - (i) XOR = A.B' + A'.B

- (ii) usando NAND-apenas (4 portas).
  Discuta portas e níveis.

8. Dada F(A,B,C,D) = A'.B' + C.D: qual solução tem menos níveis usando portas NAND-apenas? Justifique.

2) Aula Prática Laboratorial (2h) — BALF

Objetivos (Bloom):

- **Recordar:** portas, símbolos e tabelas-verdade básicas; equivalências lógicas.
- **Aplicar:** derivar funções a partir de requisitos; simplificar (álgebra/*K-map* 3–4 variáveis); implementar módulos (soma-paridade, detetor, *mux*/*decoder*).
- **Analisar:** comparar custos (portas/níveis); justificar escolhas e limites.

Folha de trabalho

Parte A — Recordar

1. Construir a tabela-verdade de F(A,B) = A $\oplus$ B.
2. Apresentar a POS canónica de F = 0 para m={1,2}.
3. Aplicar De Morgan para obter uma implementação NOR-apenas de F=(A+B)'.

Parte B — Aplicar

4. Minimize por K-map (3 variáveis): m={0,2,3,7}.
5. Derive as equações de um half-adder (S e C).
6. Projete um parity bit (par) para 3 bits d_2, d_1, d_0.

Parte C — Analisar

7. Compare duas implementações do majority-3 em número de portas e níveis (álgebra vs mux 4:1).
8. Escolha entre decoder 2→4 + portas ou mux 4:1 para realizar F(A,B,C) e justifique pela contagem de portas.

Questões

Parte A – Verdadeiro/Falso

1. Todo XOR pode ser escrito como SOP canónica.
2. Em *K-map*, grupos de 1,2,4,8… células reduzem o número de literais.
3. POS e SOP mínimas de uma função são sempre idênticas.

Parte B – Escolha múltipla

4. O número máximo de *minterms* para 4 variáveis é:
   - A) 8
   - B) 16
   - C) 32
   - D) 4
5. A forma mínima de F com m = {1,3,5,7} é:

- A) A
- B) B
- C) D
- D) C

6. De Morgan de (A.B + C)' é:
   - A) A'+B'+C'
   - B) A'+B'+C
   - C) A' + B' + C'
   - D) A'.B'.C'

Parte C – Resposta curta

12. Defina *minterm* e apresente 1 exemplo em 3 variáveis.
13. Explique por que *don't cares* podem reduzir o nº de portas.
14. Dê a equivalência XNOR em termos de AND/OR/NOT.
15. Quando preferir NAND-apenas?

Parte D – Exercícios práticos

16. Minimize a de (11) por K-map.
17. Dada F(A,B,C)=Σm(0,1,2,5,7), encontre POS mínima.
18. Realize F com *mux* 4:1 escolhendo as linhas de dados.
19. Projete *majority-3*.
20. Compare duas soluções de XOR: portas básicas vs. *mux* 2:1.
21. Mostre como implementar NOT, AND, OR usando NOR-apenas.
22. Use *don't cares* para mostrar dígitos 0–3 num 7-segmentos: simplifique o segmento a.
23. Mostre que X+X'Y=X+Y (teorema da absorção).
24. Explique como o custo em níveis afeta a latência.

3. Questões complementares

Estas questões podem ser importadas no **NotebookLM/ChatGPT** para treino extra.

Verdadeiro/Falso

1. Em *K-map* 4 variáveis, grupos de 6 são válidos.
2. *Don't cares* podem ser usados como 1 ou 0.
3. Um *half-adder* necessita de *Carry-in*.

Escolha múltipla

4. Para *majority-3*, a mínima é:
   - A) A⊕B⊕C
   - B) AB+AC+BC
   - C) ABC
   - D) A+B+C
5. O *carry* de um *full-adder* é:
   - A) AB+ACin+BCin
   - B) A⊕B⊕Cin
   - C) A'B'+Cin
   - D) AB'

6. Um *decoder* 2→4 tem:
    - A) 2 entradas, 4 saídas, 1 ativa por vez.
    - B) 4 entradas, 2 saídas.
    - C) 2 seletores.
    - D) Nenhuma.

Resposta curta

7. Escrever a tabela do *mux* 2:1.
8. Explicar como usar *mux* 4:1 para realizar qualquer F(A,B) com A,B como seletores.
9. Derivar XNOR a partir de XOR.
10. Explicar custo em portas vs. níveis.

Exercícios práticos

11. Minimizar F(A,B,C)=Σm(0,2,3,7).
12. Construir um *parity-even* para 4 bits.
13. Projetar um comparador de igualdade 2-bit.
14. Usar *decoder* 2→4 + OR para F(A,B,C)=Σm(1,3,5,7)
15. Realizar F(A,B,C)=Σm(1,4,6,7) com *mux* 2:1 (seletor A).
16. Converter F=(A+B).(A'+C) para SOP mínima.
17. Mostrar uma implementação NAND-*only* de XOR.
18. Minimizar F(A,B,C,D)=A'.B'+C.D usando portas NAND.
19. Derivar POS mínima para F=Σm(0,1,2,5,7).
20. Justificar a escolha entre mux 4:1 e lógica discreta para um dado F.

Exercícios complementares

1. Tabela-verdade do XOR
2. SOP canónica a partir de TT
3. *De Morgan* (AB+C)'=(AB)'·C'=(A'+B')C'
4. *K-map* (3 variáveis)
5. Half-adder
   S = A⊕B=A'B+AB'; C = AB
6. *Full-adder* (equações)
   S = A⊕B⊕C_{in}.
   Cout = AB+AC_{in}+BC_{in}
7. *Ripple-carry* de 2 bits
8. *Majority*-3
   Requisito: 1 se pelo menos 2 entradas são 1.
   Fórmula mínima: F=AB+AC+BC.
   Derivação: por soma de minterms 3-combinações.
9. Paridade par (3 bits)
   p=d_2⊕d_1⊕d_0.
   Prova: XOR=1 quando nº de 1's é ímpar; escolher p igual dá total par.
10. *Decoder* 2→4 (equações)
    Entradas A, B, saídas Y_0..Y_3.
    Y_0=A'B', Y_1=A'B, Y_2=AB', Y_3=AB
    Propriedade: exatamente uma saída ativa.

11. *Mux* 2:1 (função)
    Seleção S: $F=S'D_0+SD_1$.
    TT confirmando duas linhas de dados.
12. Realizar F(A,B,C)=Σm(1,4,6,7) com *mux* 2:1 (*seletor*)
    Avalie A=0: *minterms* com A=0 → apenas m1(B'C) ⇒ D0=B'C
    Avalie A=1: m4,6,7 → combinações (B,C)≠(0,1) ⇒ D1=B+C'
    Resultado:
    F=A'B'C+A(B+C')
13. Comparador de igualdade 2-bit
    Entradas A_1A_0,B_1B_0
    E (igual) = $XNOR(A_1,B_1)\cdot XNOR(A_0,B_0) = (A_1B_1+A_1'B_1').(A_0B_0+A_0'B_0')$
    Explicação: igualdade por bits e conjunção.
14. *Majority*-4 (pelo menos 3 de 4)
    Especificação: 1 se soma ≥3.
    Solução mínima (exemplo): somar combinações triplas
    ABC+ABD+ACD+BCD
    ABC+ABD+ACD+BCD.
    Comentário: pode-se usar adder + comparador ≥3.
15. *K-map* (4 variáveis) fácil
    F(A,B,C,D)=A'B'+CD.
    Implementação NAND-NAND: realize A'.B' e C.D via NAND-invertida e OR por NAND de saídas (*De Morgan*). 2 níveis lógicos.
16. XOR com 4 NAND
    Estrutura padrão:
    1.X_1=NAND(A,B'), 2) X_2=NAND(A',B) (inversões com NAND),
    2.X_3=NAND(X1,X2), 4) inverter conforme necessário.
    Validação: tabela-verdade coincide.
17. F=(A+B)(A'+C) → SOP mínima
    Expansão: F=AA'+AC+A'B+BC=0+AC+A'B+BC
    Minimização: F=AC+A'B+BC
    Por absorção BC pode ser necessário; verificar *K-map* para redundâncias.
18. "*Don't cares*" numa função de 4 variáveis
    Suponha d = {1,3,7,15}. Agrupe com 1's para ampliar blocos (8/4 células).
    Efeito: menos literais; mostrar exemplo concreto (aceitam-se soluções equivalentes).
19. Realizar F=Σm(0,1,2,5,7) com *decoder* 3→8 + OR
    Ativar saídas correspondentes e OR final.
    Comentário: escalável; custo em *fan-in* do OR.
20. Escolha tecnológica: *mux* 4:1 vs lógica discreta
    Análise: *mux* 4:1 pode reduzir níveis e facilitar mapeamento LUT (FPGA); lógica discreta pode usar menos portas em BC simples. Conclusão: justificar por contagem de portas e profundidade.

BloomArch Learning Framework

5ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**
- **20 questões por tópico (40 no total)**
- **20 exercícios laboratoriais com soluções detalhadas**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- **Recordar**: definições de tempo de execução, frequência de relógio, CPI, *throughput*, *speedup*.
- **Compreender**: relação entre tempo de execução e os três fatores fundamentais: número de instruções, CPI e frequência.
- **Aplicar**: cálculo de tempo de execução, comparação de processadores, Amdahl, métricas de desempenho.
- **Analisar**: *trade-offs* entre CPI e frequência, impacto de melhorias parciais, gargalos de desempenho.

Estratégia pedagógica (BALF)

Tempo	Atividade	Descrição
0–10 min	Diagnóstico ativo	4 perguntas VF projetadas sobre tempo de execução, CPI e frequência → discussão em grupo
10–30 min	Sessão 1 + problemas	Definições + fórmula básica: T_{CPU} = (#Instruções x CPI)/f). Resolução de 2 exemplos curtos em conjunto.
30–50 min	Sessão 2 + problemas	CPI médio ponderado, comparação entre processadores, impacto de CPI/frequência. Exercícios rápidos em pares.
50–70 min	Atividade ativa 1	Problemas de comparação de desempenho entre 2 CPU's → identificar qual é mais rápido e justificar.
70–90 min	PC-PAI (ChatGPT)	Cada estudante utiliza o *Prompt Completo* → 4–6 perguntas progressivas sobre desempenho. *Feedback* imediato e explicações.
90–110 min	Sessão 3	Introdução à Lei de Amdahl → *speedup* e limites de aceleração.
110–120 min	Fecho reflexivo	3 conceitos-chave + minute paper: "O que já consigo calcular / analisar? Onde tenho dúvidas?"

Folha de Trabalho

Parte A — Recordar

1. Defina tempo de execução de um programa.
2. Escreva a fórmula fundamental do desempenho.
3. O que significa CPI?

Parte B — Aplicar

4. Um programa executa $1{,}2\times10^9$ instruções num CPU de 2 GHz com CPI=1,2. Calcule o tempo de execução.
5. CPU A: 3 GHz, CPI=1,5; CPU B: 2 GHz, CPI=1. Qual é mais rápido para 10^9 instruções?
6. Um programa tem 3 classes de instruções com CPI = {1, 2, 4} e frequências {40%, 40%, 20%}. Calcule CPI médio.

Parte C — Analisar

7. Compare dois processadores com mesma frequência, mas CPI diferentes — qual é mais eficiente e porquê?
8. Uma melhoria acelera 40 % do programa em 4×. Qual é o *speedup* global?

2) Aula Prática Laboratorial (2h) — BALF

Objetivos:

- **Recordar:** fórmulas fundamentais (tempo de execução, CPI, frequência).
- **Aplicar:** cálculo de tempos de execução para diferentes CPUs, programas e melhorias.
- **Analisar:** identificar o processador mais rápido em diferentes cenários; aplicar Amdahl para prever *speedup* real.

Estrutura da Aula (BALF)

Tempo	Atividade	Descrição
00–10 min	*Warm-up*	3 VF curtas → *warm-up* e revisão rápida
10–35 min	Oficina 1	Cálculo de tempos de execução com diferentes CPI/frequência/nº instruções
35–60 min	Oficina 2	Cálculo de CPI médio ponderado e comparação entre CPUs
60–90 min	Oficina 3	Aplicação da Lei de Amdahl: melhorias parciais, *speedup*, limites
90–110 min	Oficina 4 (Desafio)	Análise de *trade-offs*: CPU mais rápida vs. energia / CPI
110–120 min	Fecho	Correção rápida + *error log* coletivo

Folha de Trabalho

Parte A — Recordar

1. Defina frequência de relógio e CPI.
2. Qual a relação entre frequência e período?
3. Dê um exemplo em que menos CPI não implica necessariamente menor tempo.

Parte B — Aplicar

4. Calcule o tempo de execução de 5×10^8 instruções com CPI=1,5 a 2 GHz.
5. Compare CPU X (2 GHz, CPI=2) e CPU Y (3 GHz, CPI=3). Qual é mais rápido para 10^8 instruções?
6. Um programa tem 50 % instruções tipo A (CPI=1), 30 % tipo B (CPI=2), 20 % tipo C (CPI=4). Calcule CPI médio.

Parte C — Analisar

7. Um CPU de 4 GHz executa 10^9 instruções em 0,4 s. Calcule o CPI médio.
8. Melhoria: 25 % do código acelerado em 10×. Calcule *speedup* global (Amdahl).
9. Dois CPUs têm mesmo tempo de execução, mas um tem maior frequência. O que isto implica sobre CPI?

Questões

Parte A – Verdadeiro/Falso

1. Aumentar a frequência reduz sempre o tempo de execução.
2. CPI é o número de ciclos necessário, em média, para executar uma instrução.
3. O tempo de execução depende apenas do número de instruções.
4. Dois CPUs com mesma frequência e mesmo número de instruções têm sempre o mesmo desempenho.
5. Amdahl mostra que acelerar apenas parte do programa tem ganhos limitados.

Parte B – Escolha múltipla

6. Tempo de execução depende de:
    - A) #Instr
    - B) CPI
    - C) Frequência
    - D) Todas
7. CPU A: 2 GHz, CPI=1; CPU B: 4 GHz, CPI=3. Para 10^8 instruções:
    - A) A mais rápido
    - B) B mais rápido
    - C) Empate
    - D) Nenhum
8. CPI médio ponderado calcula-se:
    - A) soma aritmética
    - B) ponderação por frequência de instruções
    - C) média geométrica

- D) nenhuma

9. Amdahl aplica-se a:
    - A) paralelismo
    - B) melhorias parciais
    - C) codificação
    - D) memória
10. Speedup global depende de:
    - A) fração melhorada
    - B) speedup local
    - C) ambas
    - D) nenhuma

Parte C – Resposta curta

11. Defina tempo de execução e dê fórmula.
12. Explique a relação frequência ↔ período.
13. Quando menor CPI pode não significar melhor desempenho?
14. Explique Amdahl com as suas palavras.
15. Dê um exemplo de melhoria local com *speedup* global limitado.

Parte D – Exercícios práticos

16. CPU: 3 GHz, CPI=2, 6×10^8 instruções → calcule T.
17. CPI médio: 3 classes CPI={1,2,4}, freq={0,5;0,3;0,2}.
18. Speedup: 30 % melhorado em 5×.
19. Comparar CPU A (2GHz, CPI=1) *vs*. B (4GHz, CPI=3).
20. Melhorias em série → combine 2 melhorias de Amdahl.

3. Questões Complementares

Estas questões podem ser importadas no **NotebookLM/ChatGPT** para treino extra.

Verdadeiro/Falso

1. A diminuição de CPI médio implica sempre um aumento proporcional do desempenho.
2. A Lei de Amdahl assume que a fração de código melhorada é independente da melhoria aplicada.
3. Um CPU com frequência mais baixa pode superar um CPU com frequência mais alta, dependendo do CPI e do número de instruções.
4. Melhorar duas frações disjuntas de um programa tem um efeito de aceleração igual à soma dos *speedups* individuais.
5. Para programas muito paralelizáveis, a Lei de Amdahl deixa de ser relevante.

Escolha múltipla

6. O que acontece ao tempo de execução se o número de instruções dobra, mantendo frequência e CPI constantes?
    - A) Reduz para metade
    - B) Aumenta 2×

- C) Permanece igual
- D) Aumenta 4×

7. Dado um CPU com 3 GHz e CPI=1, e outro com 1,5 GHz e CPI=0,4, ambos com o mesmo número de instruções, qual é mais rápido?
   - A) CPU 1
   - B) CPU 2
   - C) Iguais
   - D) Depende da ISA
8. A Lei de Amdahl mostra que:
   - A) Melhorias pequenas em frações grandes têm pouco impacto.
   - B) Melhorias grandes em frações pequenas têm pouco impacto.
   - C) Melhorias grandes em frações grandes dão pouco impacto.
   - D) Nenhuma das anteriores.
9. Num sistema, acelerar 80 % do código em 5× resulta num *speedup* máximo de:
   - A) 5×
   - B) 4×
   - C) 2,78×
   - D) 1,8×
10. Se duas melhorias afetam a mesma fração de código, qual das seguintes é verdadeira?
    - A) Os *speedups* somam-se.
    - B) Aplica-se apenas o *speedup* maior.
    - C) Aplica-se o *speedup* menor.
    - D) Aplica-se a média aritmética dos dois.

Resposta curta

11. Explique a diferença entre *speedup* local e *speedup* global, dando um exemplo numérico.
12. Por que razão a frequência de relógio deixou de ser, isoladamente, um indicador fiável de desempenho?
13. Como se calcula o CPI médio ponderado quando temos diferentes classes de instruções?
14. Um processador executa um programa em 0,5 s a 3 GHz. Qual o CPI médio se o programa tem 6×10^8 instruções?
15. Porque é que melhorias paralelas podem ter diminuição de retorno quando se combinam várias?

Exercícios práticos

1. Dois processadores A e B executam o mesmo programa:
   - A: 2,5 GHz, CPI = 1,4
   - B: 3,2 GHz, CPI = 2,1

   Número de instruções: 2×10^9

   Calcule o tempo de execução e indique qual é mais rápido.
2. Um programa tem:
   - 60 % instruções CPI=1
   - 25 % instruções CPI=2

- 15 % instruções CPI=4
  Frequência: 2 GHz, N=10^9.
  Calcule o CPI médio e tempo de execução.

3. Dois processadores têm igual frequência (3 GHz). O primeiro tem CPI=1,6 e executa $1{,}2\times10^9$ instruções. O segundo tem CPI=1,2 mas executa 2×10^9 instruções. Qual é mais rápido?
4. Melhorar 30% do código em 10×. Calcular speedup global.
5. Melhorar 90% do código em 2×. Speedup global?
6. Um programa de 10^9 instruções executa em 0,5 s a 4 GHz. Qual é o CPI médio?
7. Aceleração combinada:
   - 40% em 2×
   - 30% em 4×
     Frações disjuntas.
     Calcular *speedup* global.
8. Dois processadores:
   - X: 4 GHz, CPI=2
   - Y: 2 GHz, CPI=0,9
     Para N=5×10^8 instruções, calcule tempos.
9. Um processador com CPI=1, frequência 2 GHz é substituído por outro com CPI=1,4 e 3 GHz. Para N=10^8, qual o ganho?
10. Um programa tem duas melhorias na mesma fração de 20%:
    - Melhoria A: 2×
    - Melhoria B: 5×
      Qual aplicar?

BloomArch Learning Framework

6ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**
- **20 questões por tópico (40 no total)**
- **20 exercícios laboratoriais com soluções detalhadas**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- **Recordar:** identificar os principais componentes da organização do computador (CPU, ALU, UC, registos, memória, E/S, barramentos).
- **Compreender:** explicar o papel de cada componente e o fluxo de dados/controlos.
- **Aplicar:** analisar instruções simples e descrever como percorrem o *datapath*.
- **Analisar:** raciocinar sobre gargalos e otimizações básicas (e.g., utilização de registos vs memória).

Estratégia pedagógica (BALF)

Tempo	Atividade	Descrição
0–10 min	Diagnóstico ativo	4 perguntas VF projetadas sobre CPU/memória/barramentos.
10–30 min	Explicação 1 + micro-problemas	Organização de alto nível: CPU–memória–E/S; barramento único vs múltiplos; localização dos registos e UC.
30–50 min	Explicação 2 + problemas guiados	Caminho de dados: execução de instruções do ciclo *fetch–decode–execute*.
50–70 min	Atividade ativa	Em pares: desenhar um datapath simples e explicar o trajeto de uma instrução ADD R1, R2, R3.
70–90 min	PC-PAI (ChatGPT)	Autoestudo guiado: 5 perguntas interativas (VF, MC, curta, prática) sobre organização.
90–110 min	Explicação 3	UC: controlo por cablagem *vs*. microprogramado. Exemplos curtos.
110–120 min	Fecho reflexivo	“Três conceitos-chave + dúvida mais frequente” — plenário rápido.

Folha de Trabalho

Parte A — Recordar

1. Listar os principais componentes da organização de um computador.
2. O que é o *datapath*?
3. Qual o papel da Unidade de Controlo?

Parte B — Aplicar

4. Desenhar e rotular um esquema simplificado CPU–Memória–E/S (com barramento).
5. Descrever as etapas do ciclo *fetch–decode–execute* para uma instrução LOAD R1, 100(R2).
6. Explicar como o barramento é usado numa operação de leitura de memória.

Parte C — Analisar

7. Explicar a diferença entre registos e memória principal em termos de velocidade e utilização.
8. Comparar controlo por cablagem e microprogramado quanto a flexibilidade e desempenho.

2) Aula Prática Laboratorial (2h) — BALF

Objetivos:

- **Recordar**: sintaxe básica (formato R/I/J), registos, instruções aritméticas, *load/store*, *branch*.
- **Aplicar**: escrever programas simples em Assembly; usar registos corretamente; converter entre alto nível e Assembly.
- **Analisar**: otimizar trechos de código, identificar padrões de instruções, analisar uso de registos *vs*. memória

Estrutura da Aula (BALF)

Tempo	Atividade	Descrição
0–15 min	*Warm-up* ativo	Tradução de 3 instruções básicas (R e I-type).
15–40 min	Oficina 1	Conversão de pseudo-código para MIPS: aritmética básica, load/store.
40–65 min	Oficina 2	Branches e jumps: tradução de if/else e loops simples.
65–90 min	Oficina 3	PC-PAA (ChatGPT) — questionário assembly com *feedback* imediato.
90–115 min	Oficina 4	Programa completo: soma de vetor, contador condicional ou multiplicação por deslocamento.

115–120 min	Fecho	Revisão rápida + erros comuns.

Folha de Trabalho

Parte A — Recordar

1. Indicar o formato das instruções R, I e J.
2. Quais os registos reservados (e.g., $zero, $ra, $sp)?
3. Explicar a diferença entre lw e sw.

Parte B — Aplicar

4. Traduzir C = A + B - D assumindo A, B, D, C em $s0–$s3.
5. Traduzir if (R1 == R2) R3 = R3 + 1; else R3 = R3 - 1;
6. Implementar um ciclo for(i=0; i<10; i++) s=s+i;

Parte C — Analisar

7. Reescrever uma soma de vetor minimizando acessos à memória.
8. Identificar padrões de dependências RAW e explique como resolvê-las (reordenação / NOPs).

Questões

Parte A – Verdadeiro/Falso

1. A unidade de controlo (UC) é responsável por executar operações aritméticas e lógicas.
2. Os registos são normalmente mais rápidos do que a memória principal.
3. O barramento de endereços transporta dados entre CPU e memória.
4. O ciclo *fetch–decode–execute* descreve o processo fundamental de execução de instruções.
5. Num sistema com barramento único, apenas uma transferência pode ocorrer de cada vez.

Parte B – Escolha múltipla

7. Qual dos seguintes NÃO é um componente principal da organização de um computador?
    - A) CPU
    - B) Memória
    - C) Compilador
    - D) Módulos de E/S
8. Qual das opções representa corretamente as três etapas do ciclo de instrução?
    - A) pesquisar, executar, compilar
    - B) pesquisar, decodificar, executar
    - C) ler, escrever, compilar
    - D) interpretar, executar, armazenar
9. Qual das seguintes afirmações sobre a unidade de controlo microprogramada é correta?

- A) É mais rápida que a cablada
- B) É mais flexível e fácil de modificar
- C) Não utiliza memória de controlo
- D) Não pode implementar instruções complexas

10. O que transporta o barramento de controlo?
    - A) Endereços
    - B) Dados
    - C) Sinais de sincronização e comandos
    - D) Instruções
11. Qual das seguintes estruturas é normalmente mais próxima do ALU?
    - A) Memória cache
    - B) Memória principal
    - C) Registos
    - D) Barramentos

Parte C – Resposta curta

12. Definir “datapath” num processador.
13. Explicar, em 2-3 frases, o papel da Unidade de Controlo.
14. Descrever brevemente as diferenças entre controlo por cablagem e microprogramado.
15. Qual é a função do barramento de endereços?
16. Porque é que os registos são usados em vez da memória sempre que possível?

Parte D – Exercícios práticos

17. Desenhar (ou descrever verbalmente) um esquema simples da organização CPU–Memória–E/S com barramento único e explique como uma instrução LOAD R1, 100(R2) é executada.
18. Listar e explicar as etapas do ciclo fetch–decode–execute para uma instrução ADD R3, R1, R2.
19. Comparar um sistema de barramento único com um sistema de múltiplos barramentos em termos de desempenho.
20. Numa arquitetura microprogramada, como se altera uma instrução complexa sem mudar o hardware?
21. Explicar como os sinais de controlo coordenam as seguintes ações numa leitura de memória.

3. Questões Complementares

Estas questões podem ser importadas no **NotebookLM/ChatGPT** para treino extra.

Verdadeiro/Falso

1. Em MIPS, beq compara dois registos e salta se forem iguais.
2. MIPS é load–store: instruções ALU podem ler/escrever diretamente memória.
3. O registo $zero pode ser modificado por uma instrução addi

4. Instruções J-type usam um alvo de 26 bits que é concatenado com bits superiores de PC+4 (com shift de 2).
5. lw $t0, 12($s1) lê a word em endereço [$s1 + 12] para $t0.

Escolha múltipla

6. Qual é R-type?
    - A) addi
    - B) lw
    - C) and
    - D) ori
7. O endereço efetivo de lw $t0, 12($s1) é:
    - A) 12 - $s1
    - B) $s1 << 12
    - C) $s1 + 12
    - D) $t0 + 12
8. Para retornar de uma função chamada com jal, usa-se:
    - A) jr $ra
    - B) j $ra
    - C) jalr $ra.
    - D) ret
9. Em beq $t0,$t1,label, o offset imediato representa:
    - A) bytes a partir do PC atual
    - B) palavras (word offsets) relativas a PC+4
    - C) páginas de memória
    - D) endereço absoluto
10. Qual sequência implementa “x = x + (a < b ? 1 : -1)” assumindo x=$s0, a=$s1, b=$s2?
    - A) slt $t0,$s1,$s2; addi $s0,$s0,1; beq $t0,$zero,end; addi $s0,$s0,-2; end:
    - B) slt $t0,$s2,$s1; addi $s0,$s0,1; beq $t0,$zero,end; addi $s0,$s0,-2; end:
    - C) slt $t0,$s1,$s2; addi $s0,$s0,-1; bne $t0,$zero,end; addi $s0,$s0,2; end:
    - D) slt $t0,$s1,$s2; addi $s0,$s0,-1; beq $t0,$zero,end; addi $s0,$s0,2; end:

Resposta curta

11. Diferença entre pseudoinstrução e instrução real (com exemplo).
12. Traduzir x = 4*x + 3 assumindo x=$s0 (sem multiplicação).
13. Como calcular o endereço de A[i] (inteiros 4 bytes), com A em $s0 e i em $t0?
14. Identificar o *hazard* em:

```
lw  $t0,0($s0)
add $t1,$t0,$t2
```

15. O que fazem jal e jr $ra?

Exercícios práticos

1. Converter o if–else em Assembly (sem slt):

   if (a == b) x = x + 1; else x = x - 1;

   Assumir a=$s1, b=$s2, x=$s0.

2. Soma de vetor (endereçamento por apontadores):

   s = 0; for(i=0;i<n;i++) s += A[i];

   Assumir A=$s0 (base), n=$s1, s=$s2.

3. Máximo de vetor: devolve max em $s2. Assumir n>0, A=$s0, n=$s1
4. Função sumArray(int* A, int n) que retorna soma ($v0), com prologue/epilogue.
5. Traduzir o while sem slt (usar bne/beq e aritmética):

   i = 0; s = 0;

   while (i < n) { s += i*i; i++; }

   Assumir i=$s0, s=$s1, n=$s2.

BloomArch Learning Framework

7ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- Avaliar, através de um teste, a consolidação dos conteúdos das semanas 1–6:
  - Organização física do computador
  - Representação da informação
  - Circuitos lógicos
  - Circuitos combinatórios
  - Noções de desempenho de processadores
- Integrar níveis de Bloom: recordar, aplicar, analisar.

Possível estrutura sugerida do exame (2 h)

Parte	Tipo de Questões	Nº Questões	Pontos
A	Verdadeiro/Falso	5	10%
B	Escolha múltipla	5	20%
C	Resposta curta	1	20%
D	Problemas analíticos	6	50%

2) Aula Prática Laboratorial (2h) — BALF

Objetivos:

- **Recordar**: formatos e instruções fundamentais.
- **Aplicar**: tradução de estruturas de controlo mais complexas e manipulação de dados em Assembly.
- **Analisar**: otimizar código Assembly, identificar dependências, estruturar programas não triviais.

Estrutura da Aula (BALF)

Tempo	Atividade	Descrição
0–15 min	*Warm-up* ativo	Tradução de instruções e estruturas condicionais intermédias (em grupo).
15–45 min	Oficina 1	Tradução de estruturas *while* aninhadas, *switch*, contagens condicionais.

45–75 min	Oficina 2	PC-PAA no ChatGPT → questionário interativo de assembly com feedback imediato sobre as soluções criadas pelo estudante em comparação com as soluções gerados pelo ChatGPT.
75–115 min	Oficina 3	Programas completos com múltiplos *loops* e acessos à memória.
115–120 min	Fecho	Revisão rápida dos conceitos trabalhados.

Folha de Trabalho

Parte A — Recordar

1. Qual é o formato de uma instrução beq?
2. Qual é a diferença entre jal e jr $ra?
3. Qual a utilidade do registo $ra em chamadas de função?

Parte B — Aplicar

4. Traduza um ciclo while aninhado (duplo loop) para Assembly.
5. Implemente um switch simples usando branch e jump.
6. Escreva um programa que conte o número de elementos negativos num vetor.

Parte C — Analisar

7. Identifique dependências RAW e explique como reordenar instruções para as minimizar.
8. Otimize um código assembly dado, reduzindo loads redundantes e reorganizando instruções.

Questões

Parte A – Verdadeiro/Falso

1. A instrução lw transfere dados de um registo para a memória.
2. A instrução jal guarda o endereço de retorno no registo $ra.
3. O formato da instrução beq é do tipo R.
4. O registo $zero pode ser modificado por uma instrução addi.
5. A instrução jr $ra é usada para retornar de uma sub-rotina.

Parte B – Escolha múltipla

6. Qual das seguintes instruções usa o formato J-type?
    - A) add
    - B) beq
    - C) j
    - D) lw
7. Qual instrução é usada para chamar uma sub-rotina?
    - A) jal
    - B) jr
    - C) beq
    - D) addi

8. Qual o papel do registo $ra?
    - A) Contém dados temporários
    - B) Guarda o endereço base de um vetor
    - C) Contém endereço de retorno de uma chamada de função
    - D) Guarda o endereço de pilha atual
9. Se quisermos comparar dois registos e saltar se forem diferentes, usamos:
    - A) beq
    - B) bne
    - C) j
    - D) addi
10. Qual destas instruções não acede à memória?
    - A) lw
    - B) sw
    - C) add
    - D) lui

Parte C – Resposta curta

11. Explicar a diferença entre lw e sw.
12. O que faz jal Label?
13. Como se calcula o endereço efetivo numa instrução I-type de acesso à memória?
14. Qual é o conteúdo do registo $zero? Pode ser alterado?
15. Explique a função de jr $ra numa sub-rotina.

Parte D – Exercícios práticos

16. Tradução de if-else. Traduzir para Assembly:

```
if (x > y) z = x;
else z = y;
```

    Assumindo: x=$s0, y=$s1, z=$s2
17. Traduzir:

```
for (i=0; i<10; i++) s = s + i;
```

    Assumindo: i=$t0, s=$s0
18. Soma de vetor. Enunciado: Somar N elementos de vetor A, resultado em $s0.
19. Sub-rotina soma de dois vetores

```
void somaVetores(int *A, int *B, int *C, int n)
{
  for (i=0;i<n;i++) C[i]=A[i]+B[i];
}
```

    Assumindo: A0=A, A1=B, A2=C, A3=n
20. Análise de *hazards* e otimização. Analise:

```
lw $t0, 0($s0)
add $t1, $t0, $t2
lw $t3, 4($s0)
add $t4, $t3, $t5
```

3. Questões Complementares

Estas questões podem ser importadas no **NotebookLM/ChatGPT** para treino extra.

Verdadeiro/Falso

1. A instrução sw escreve um valor de um registo para a memória.
2. Em MIPS, a instrução beq tem formato R-type.
3. O registo $ra armazena o endereço de retorno de uma chamada a sub-rotina (jal).
4. Um mesmo endereço pode ser carregado múltiplas vezes com lw sem impacto de desempenho.
5. As dependências RAW (Read After Write) podem causar stalls no pipeline se não forem tratadas.

Escolha múltipla

6. Qual destas instruções carrega dados da memória para um registo?
    - A) sw
    - B) add
    - C) lw
    - D) beq
7. Qual destas instruções pertence ao formato J-type?
    - A) add
    - B) beq
    - C) jal
    - D) addi
8. Qual é o registo normalmente usado para armazenar o endereço de retorno?
    - A) $t0
    - B) $ra
    - C) $a0
    - D) $sp
9. Qual é a sequência correta para implementar um if (x==y) em MIPS?
    - A) bne, depois salto para etiqueta
    - B) beq, depois salto condicional para etiqueta “then”
    - C) j, depois comparação
    - D) sw, depois branch
10. Para traduzir um for em Assembly MIPS, são necessárias pelo menos:
    - A) 1 instrução (beq)
    - B) 2 instruções (li, j)
    - C) Inicialização + comparação + incremento + corpo
    - D) Nenhuma

Resposta curta

11. Explicar a diferença entre jal e jr $ra.
12. O que é uma dependência RAW?
13. Por que razão se devem evitar loads redundantes?
14. O que faz a instrução slt $t0,$t1,$t2?
15. Descrever como traduzir uma condição composta: while (i<n && x>0) { ... }

Exercícios práticos

1. Tradução if-else. Traduzir:

   if (x > y) z = x;

   else z = y;

2. Loop For. Traduzir:

   for (i=0; i<5; i++) s = s + i;

3. Contar números negativos num vetor
4. Sub-rotina soma de vetor. Crie uma sub-rotina que recebe base e N, devolve soma.
5. Otimização com hazards. Excerto original:

   lw $t0, 0($s0)

   add $t1, $t0, $t2

   lw $t3, 4($s0)

   add $t4, $t3, $t5

BloomArch Learning Framework

8ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**
- **20 questões por tópico (40 no total)**
- **5 exercícios laboratoriais com soluções detalhadas**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- **Compreender** o princípio da localidade temporal e espacial.
- **Distinguir** *cache* mapeada diretamente, associativa e associativa por conjunto.
- **Calcular** taxa de acertos (*hit ratio*), tempo médio de acesso e penalidade de falha.
- **Analisar** o impacto do tamanho da linha, número de conjuntos e política de substituição no desempenho.
- **Relacionar** *cache* L1, L2, L3 e hierarquias de memória reais (ex: Intel, ARM).

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Breves explicações intercaladas com pequenos problemas de cálculo e interpretação de mapas de *cache*.
PC-PAI no ChatGPT	Autoestudo interativo com perguntas geradas sobre os ficheiros do Stallings (Cap. 4).
Feedback imediato	Discussão dialogada após cada exercício.
Progressão Bloom	Recordar → Aplicar → Analisar (com problemas quantitativos e qualitativos).

Folha de Trabalho

Parte A — Recordar

1. Defina localidade temporal e localidade espacial.
2. Descreva as três políticas de mapeamento de *cache*.
3. Explique o significado de hit e miss.
4. Dê um exemplo de *write-through* e *write-back*.
5. Indique as vantagens de caches multinível (L1, L2, L3).

Parte B — Aplicar

6. Uma cache de 4 KB, linhas de 16 B e memória de 64 KB.
   - Calcule o nº total de linhas.
   - Determine quantos *bits* são usados para *offset*, índice e *tag*.
7. Um programa tem 1000 acessos à memória, com 900 acertos.
   - Qual é o *hit ratio* e o *miss ratio*?
   - Se o tempo de acesso à cache é 5 *ns* e à memória principal é 50 *ns*, determine o tempo médio de acesso.
8. Numa *cache* associativa por conjunto com 4 conjuntos e 2 vias, quantas linhas existem no total?
9. Dada uma sequência de endereços, determine quais resultam em *miss* para uma *cache* de mapeamento direto.

   Endereços: 0, 4, 8, 0, 16, 0

   Cache: 2 linhas, 4 bytes/linha
10. Explique, com cálculo, o impacto do aumento do bloco de 4 B para 32 B na taxa de acertos.

Parte C — Analisar

11. Compare *write-through* e *write-back* em termos de coerência e latência.
12. Como a política de substituição LRU influencia o desempenho?
13. Por que motivo *caches* maiores nem sempre são mais rápidas?
14. Compare hierarquias modernas (Intel Core vs ARM).
15. Explique o impacto de *prefetching*.

2. Aula Prática Laboratorial (2h)

Objetivos:

- **Reforçar** a tradução de instruções.
- **Usar** o MARS para testar resultados.
- **Aplicar** PC-PAA no ChatGPT para compreender passo a passo cada instrução.

Estrutura da Aula (BALF)

Tempo	Atividade	Objetivos
0–10 min	Introdução e recapitulação	Revisão das instruções básicas (add, sub, lw, sw).
10–25 min	Explicação e pequenos problemas: operações lógicas	Compreender efeitos de AND, OR, NOR, XOR.
25–45 min	Exercício 1 da Ficha6: implementação e tabela de resultados	Aplicar operações bit a bit.

45–60 min	Explicação: instruções de deslocamento (sll, srl, sra)	Entender diferença entre deslocamento lógico e aritmético.
60–80 min	Exercício 2: aplicação de *shifts* e máscaras	Manipular *bits* e extrair *nibbles*.
80–95 min	Exercício 3: diretivas do assembler e *syscalls*	Trabalhar .data, .text, .asciiz, la, print_str, read_int.
95–110 min	Sessão com ChatGPT (PC-PAA)	Autoavaliação: o aluno descreve o que aprendeu, testa variantes e pede feedback ao ChatGPT.
110–120 min	Fecho e reflexão	Debate coletivo sobre erros comuns, otimizações e aprendizagens.

Folha de Trabalho

Parte A — Recordar

1. Dado $t0=0x12345678 e $t1=0x0000FF00, determine (sem executar no simulador):
   a) $t2 = $t0 AND $t1
   b) $t3 = $t0 OR $t1
   c) $t4 = $t0 XOR $t1
   d) $t5 = NOR($t0, $t1)
2. Para cada caso, calcule sll, srl e sra de 4 bits:
   - $t0 = 0x12345678 (positivo)
   - $t0 = 0x862A5C1B (negativo: MSB=1)
3. Escreva um programa mínimo que imprima a *string* “AOC - Semana 8” seguida de nova linha. Use .data, .text, .asciiz, la, e syscalls adequadas.

Parte B — Aplicar

4. Traduza:
   if (x < y) z = x + 1;
   else z = y - 1;
5. Some os N elementos de A e guarde a soma em $s3. Assuma: base(A) em $s0, N em $s1.
6. Dado um inteiro em $t0, imprima os 8 dígitos hex do mais significativo para o menos significativo.

Parte C — Analisar

7. No trecho abaixo há *hazards* de *load-use*. Reescreva para evitar stall sem alterar o resultado:
   ```
   lw  $t0, 0($s0)
   add $t1, $t0, $t2
   lw  $t3, 4($s0)
   add $t4, $t3, $t5
   ```
8. Otimizar removendo redundâncias e reduzindo dependências:
   ```
   lw  $t0, 0($s0)
   ```

```
add $t1, $t0, $t2
lw  $t0, 0($s0)     # redundante
sub $t3, $t0, $t4
sw  $t3, 12($s0)
```

9. Reescrever o *loop* para melhorar localidade e reduzir custo de controlo. Mostrar desenrolamento x2 (*loop unrolling*).

Exercícios

1. Sem executar no simulador, determina os resultados (em hex) e depois confirma em MARS:
   Dado $t0 = 0x00FF_3300 e $t1 = 0x0F0F_00F0, calcula:
   $t2 = $t0 AND $t1
   $t3 = $t0 OR $t1
   $t4 = $t0 XOR $t1
   Cria uma máscara para limpar o byte menos significativo de $t0 e forçar o *nibble* alto do byte mais significativo a 0xA (isto é, resultado tem a forma 0xA?_____).

2. Sem correr o simulador, calcula (em hex) para 4 bits de deslocamento:
   a) $t0 = 0x3124_8001 → sll 4, srl 4, sra 4
   b) $t0 = 0xC125_000F → sll 4, srl 4, sra 4

3. Escrever um programa que imprima apenas os *nibbles* de índice ímpar do registo $a0 (32 bits), do mais significativo para o menos significativo. Exemplo: se $a0 = 0xDEAD_BEEF, imprime: E-D-B-E (nibbles #7,#5,#3,#1). Usa *lookup* "0123456789ABCDEF" e print_char (syscall 11).

4. Ler um inteiro N do utilizador. Depois ler N inteiros e contar quantos têm o *bit* 15 (contando do 0) a 1. Imprimir o total.

5. Enunciado
   1. Implementar uma sub-rotina rol32(x,k) que executa 32-bit à esquerda k posições (0..31). Interface:
      - Entrada: $a0=x, $a1=k
      - Saída: $v0 = (x << k) | (x >> (32-k)) (com *shifts* lógicos variáveis: sllv/srlv)
   2. Dado um vetor de N inteiros (base em $s0, N em $s1), criar um ciclo que aplica rol32(valor,8) a cada elemento e conta quantos resultados têm o MSB=1.
   3. Otimizar o ciclo com unrolling x2 e explica como foi evitado load-use stalls.

Como aplicar o PC-PAA:

- Para os exercícios (1-5), escrever a solução e pedir ao ChatGPT para gerar uma solução. Aplicar o PC-PAA da forma descrita para se observar as diferenças.
- Exercícios 1

- Pedir ao ChatGPT: "*Verifica os meus cálculos AND/OR/XOR e explica por que a operação 4 precisa de duas máscaras (limpar + forçar)*".
- Deve pedir perguntas guiadas: "*Dá-me três variações onde o byte alvo não seja o LSB*".
- Deve solicitar explicação de erros: "*Se eu usar só OR com 0xA0000000 sem limpar antes, o que pode acontecer?*".

- Exercício 2
  - "Explicar, com desenho de bits, por que sra difere de srl no caso b)."
  - "Dá-me 2 valores extra (um positivo e um negativo) e pede-me para prever os três resultados antes de correr."
  - "Onde posso errar ao interpretar sra com valores negativos? Testa-me com perguntas V/F."
- Exercício 3
  - Explica a minha lógica para selecionar apenas nibbles ímpares. Há forma mais simples (p. ex., um contador que faz i=i-2)?"
  - "Pede-me para adaptar o programa a imprimir apenas nibbles pares (#6,#4,#2,#0)."
  - "Avalia o meu código: posso trocar srlv por srl? Em que casos?"
- Exercício 4
  - "Verificar se testei o bit certo e propõe variações (ex.: bit 23, bit 0)."
  - "Mostrar como minimizar stalls se eu também guardasse os N valores num vetor (sugere ordem de lw/processamento)."
  - "Avaliar complexidade em ciclos e sugere micro-otimização (p. ex., ler dois ints antes de testar para reduzir branches)."
- Exercício 5
  - "Verificar a correção da minha rol32 com casos extremos: k=0, k=31."
  - "Ajuda-me a reescrever o laço com menos dependências (sugerir ordem de loads/calls/testes)."
  - "Gera perguntas de autoavaliação (básico→avançado) sobre rotate vs shift e a influência no MSB."
  - "Pede-me para medir (estimativa) o nº de instruções por iteração com e sem unrolling."

Notas finais de estudo autónomo com o PC-PAA

1. **Alternância**: tenta prever resultados (papel) → valida no MARS → pede ao ChatGPT para explicar diferenças.
2. **Socraticamente**: pede ao ChatGPT para não dar o código final de imediato — solicitar pistas graduais.
3. **Reflexão**: no final, solicita um resumo das dificuldades e um plano de treino (3 exercícios novos criados pelo ChatGPT, baseados nos teus erros).

BloomArch Learning Framework

9ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**
- **20 questões por tópico (40 no total)**
- **5 exercícios laboratoriais com soluções detalhadas**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- **Compreender** estrutura da memória principal
- **Comparar** SRAM vs DRAM
- **Diferençar** módulos SIMM, DIMM, canais
- **Conhecer** paridade e ECC
- **Analisar** impacto da latência e largura de banda

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Cada explicação teórica é imediatamente seguida de pequenos exercícios curtos no quadro e quiz oral com toda a turma
PC-PAI no ChatGPT	Autoestudo interativo com perguntas geradas sobre o capítulo do Stallings (Cap. 5).
Feedback imediato	Discussão dialogada após cada exercício.
Progressão Bloom	Progressão gradual: Recordar (conceitos SRAM/DRAM e módulos), Aplicar (cálculos de latência e largura de banda), Analisar (impacto real no desempenho do sistema)

Tempo	Atividade	Objetivo	Bloom
0–10 min	Revisão e diagnóstico via PC-PAI	Identificar lacunas iniciais	Recordar
10–25 min	Mini-aula: SRAM vs DRAM	Interpretar função e construção	Compreender
25–30 min	Pequenos exercícios guiados	Fixação	Aplicar
30–45 min	Módulos DIMM, paridade, ECC	Analisar proteção de dados	Compreender/Analisar
45–60 min	Cálculos de latência e largura de banda	Quantificar desempenho	Aplicar

60–75 min	Comparação prática CL/Freq (*benchmarks* teóricos)	Tomada de decisão técnica	Analisar
75–95 min	PC-PAI — sessão individual orientada	Autonomia e reflexão	Aplicar/Analisar
95–110 min	Discussão conjunta dos resultados do PC-PAI	Metacognição	Avaliar
110–120 min	Síntese + questões abertas para a próxima aula	Consolidação do conhecimento	Criar/Planear

Folha de Trabalho

Parte A — Recordar

1. Definir SRAM e DRAM. Dar um exemplo de onde são utilizadas no computador.
2. O que é a paridade? Em que difere de ECC?
3. O que significa tempo de acesso? Por que é crítico?
4. O que é um módulo DIMM? Compare com SIMM.
5. O que é "*dual channel*" na memória RAM?

Parte B — Aplicar

6. Explicar o papel do *refresh* na DRAM e descrever o seu impacto no desempenho.
7. A memória tem largura de banda de 25 GB/s. Quantos *bytes* consegue transferir em 10 *ms*?
8. Um sistema tem 8 *GB* de RAM DDR4-3200. Qual é o débito efetivo de transferência por módulo?
9. Dado um acesso com latência CAS = 16 ciclos numa frequência de 1600 *MHz*, calcular a latência em *ns*.
10. Comparar dois módulos e indicar qual é melhor e porquê:
    A) DDR4-2666, CL19
    B) DDR4-3200, CL22

Parte C — Analisar

11. Justificar tecnicamente porque o CPU utiliza *cache* SRAM em vez de DRAM.
12. Relacionar *single rank* vs *dual rank* com paralelismo interno.
13. Qual o impacto de ECC em servidores? Quando não se recomenda?
14. Se um sistema tem um estrangulamento na memória, quais intervenções de *hardware* melhoram o desempenho?
15. Explicar por que DDR significa *Double Data Rate* e como isso altera o cálculo do débito real.

2. Aula Prática Laboratorial (2h)

Objetivos:

- **Aplicar saltos condicionais e incondicionais** (beq, bne, j, jal, jr)
- **Implementar if/else, while, for, switch-case**
- **Utilizar funções/sub-rotinas** com convenções de chamada ($ra, $sp, $a*, $v*)
- **Criar** e interpretar **fluxogramas → Assembly**
- **Detetar** e solucionar ***hazards* lógicos** por controlo
- **Aplicar** PC-PAA no ChatGPT para compreender passo a passo cada instrução.

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Pequenas tarefas e simulação contínua com MARS
PC-PAA no ChatGPT	Autoavaliação: o estuante descreve o que aprendeu, testa variantes e pede *feedback* ao ChatGPT
Feedback imediato	Discussão dialogada após cada exercício.
Progressão Bloom	Recordar → Aplicar → Analisar fluxos de controlo e execução

Folha de Trabalho

Parte A — Recordar

1. Escrever uma condição simples em Assembly MIPS:
   if (x == y) x = 0;
2. Traduzir:
   while (i > 0) { i--; }
3. Indicar o objetivo de jal e de jr $ra.

Parte B — Aplicar

4. Traduzir para Assembly MIPS:
   if (a >= b) a = a - b;
   else a = b - a;
5. Traduzir para MIPS:
   for (i = 10; i > 0; i--) soma += i;
6. Implemente um *switch-case*, com 3 casos e *default* em Assembly MIPS:
   switch(op){
     case 0: res=1; break;
     case 1: res=2; break;
     default: res=0;
   }

Parte C — Analisar

7. Indicar potenciais dependências RAW no seguinte código e reordene:
```
lw $t0, 0($s0)
add $t1, $t0, $t2
lw $t3, 4($s0)
add $t4, $t3, $t5
```
8. Explicar o que acontece se nunca guardarmos $ra antes de chamar recursivamente uma função.
9. Otimizar este código para reduzir *overhead* de controlo:
```
for (i = 0; i < 4; i++){
   soma += A[i];
}
```

Exercícios

1. *strcmp* em Assembly MIPS com três retornos (-1, 0, +1). Implementar a sub-rotina:
```
int my_strcmp(const char* s1, const char* s2);
/* retorna -1 se s1<s2, 0 se iguais, +1 se s1>s2 (ASCII) */
```
   - Argumentos: $a0=s1, $a1=s2
   - Retorno: $v0 ∈ {-1,0,1}
   - Percorrer ambas as *strings* até encontrar diferença ou '\0'.

2. *switch* com *jump table* (tabela de saltos computados). Implementar em Assembly MIPS:
```
int op_select(int x, int y, int op);
/* op ∈ {0,1,2,3}:
  0 -> x+y
  1 -> x-y
  2 -> x*y
  3 -> (y==0 ? 0 : x/y)  // divisão inteira segura
  default -> -1 (proteção)
*/
```
   - $a0=x, $a1=y, $a2=op, retorna em $v0.
   - Usar tabela de saltos: jr a um endereço obtido de uma tabela de *labels*.

3. Fatorial recursivo com *stack frame* correto + versão iterativa (otimizada). Implementar em Assembly MIPS int fact(int n) com recursão:
   - Caso base: n<=1 → 1
   - Caso geral: n * fact(n-1)

   Em seguida, mostra versão iterativa e compara o custo de chamadas.
4. Pesquisar a primeira ocorrência > T numa matriz. Dada uma matriz A de R linhas por C colunas (inteiros 32-bit, *row-major*), encontrar a primeira posição (i,j) tal que A[i][j] > T.
   - Entradas: $a0 = baseA, $a1 = R, $a2 = C, $a3 = T
   - Saídas: $v0 = i e $v1 = j, ou -1, -1 se não existir.

5. Soma + Contagem de positivos com *unrolling* ×4 e *scheduling anti-stall.* Dado um vetor A com N inteiros, calcular:
   - a soma total em $s2;
   - o número de positivos em $s3.

   Implementar com desenrolamento x4 e agendamento de instruções para reduzir *load-use*.

Como aplicar o PC-PAA (autoestudo).

- Para os exercícios (1-5), escrever a solução e pedir ao ChatGPT para gerar uma solução. Aplicar o PC-PAA da forma descrita para se observar as diferenças.
- Exercício 1
  - Pedir variação: "*E se eu quiser valor de retorno c1-c2 (C clássico), como mudo o fluxo?*"
  - Solicitar inspeção: "*Mostra onde poderia surgir bug se usasse lb em vez de lbu.*"
- Exercício 2
  - "*Gera testes aleatórios para cada op e verifica o retorno esperado.*"
  - "*Põe-me a converter esta implementação para beq em cadeia (sem jump table) e explica o impacto.*"
  - "*Avalia riscos de mul/div (overflow, divisão por zero) e desafia-me a tratá-los.*"
- Exercício 3
  - "*Verifica que na versão recursiva restauro $ra/$sp em todos os caminhos.*"
  - "*Pede-me para medir instruções/chamada (estimativa) e comparar com o iterativo.*"
  - "*Desafia: adicionar deteção simples de overflow (se $v0 ficar negativo → sinaliza erro).*"
- Exercício 4
  - "Gera arrays de teste (inclui casos sem solução) e verifica os índices devolvidos."
  - "Desafia: inverter a ordem da pesquisa (por colunas) e discutir a localidade."
  - "Pede sugestões para minimizar custos de cálculo do endereço (ex.: manter ptr da linha e só somar 4 a cada coluna)."
- Exercício 5
  - "*Pede análise de pipeline: onde ainda podem existir stalls? Dá-me uma versão com menos nop.*"
  - "*Sugere métricas: quantos branches/elemento antes e depois do unrolling?*"
  - "*Desafia: adaptar o unrolling ×8 e estimar o impacto na I-cache.*"

Notas finais de estudo autónomo com o PC-PAA

1. **Alternância**: tenta prever resultados (papel) → valida no MARS → pedir ao ChatGPT para explicar as diferenças.

2. **Socraticamente**: pede ao ChatGPT para não dar o código final de imediato — solicitar pistas graduais**.**
3. **Reflexão**: no final, solicita um resumo das dificuldades e um plano de treino (3 exercícios novos criados pelo ChatGPT, baseados nos teus erros).

BloomArch Learning Framework

10ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**
- **20 questões por tópico (40 no total)**
- **5 exercícios laboratoriais com soluções detalhadas**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- **Compreender** o conceito de hierarquia de memória e a sua justificação económica e de desempenho.
- **Distinguir** memória primária, *cache* e secundária, analisando tempos de acesso e custos por *bit*.
- **Identificar** tecnologias de armazenamento (SSD, HDD, Flash, MRAM, etc.).
- **Calcular** médias de tempo de acesso **e** eficiências globais da hierarquia.
- **Analisar** a interação entre processador, *cache* e memória secundária.

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Explicações intercaladas com cálculos e exemplos de tempo de acesso
PC-PAI no ChatGPT	Autoestudo interativo com perguntas geradas sobre o capítulo do Stallings (Cap. 6).
Feedback imediato	Correção coletiva e automática através da IA.
Progressão Bloom	Progressão: recordar → aplicar → analisar (Aplicação em problemas com hierarquia 2 e 3 níveis)

Tempo	Atividade	Objetivo	Nível Bloom
0–10 min	Diagnóstico com PC-PAI	Identificar conhecimentos prévios	Recordar
10–35 min	Explicação intercalada com pequenos exercícios	Hierarquia e tempos de acesso	Compreender
35–60 min	Cálculo aplicado	Eficiência global da hierarquia	Aplicar
60–90 min	Casos reais (SSD vs HDD vs RAM)	Comparar tecnologias	Analisar

90–120 min	Sessão PC-PAI individual	Consolidar e autoavaliar	Aplicar/Analisar

Folha de Trabalho

Parte A — Recordar

1. O que é uma hierarquia de memória e porque existe?
2. Qual é a principal diferença entre DRAM e Flash?
3. Definir “tempo médio de acesso”.
4. O que significa “taxa de acerto na *cache*”?
5. Identificar os principais tipos de armazenamento secundário.

Parte B — Aplicar

6. Um sistema tem *cache* (hit = 95%, 2 ns) e RAM (miss = 10 ns). Calcular o tempo médio de acesso.
7. Numa hierarquia de três níveis (L1, L2, RAM), os tempos são: 1 ns, 5 ns e 50 ns, com taxas de acerto de 90% e 95%. Calcular o tempo médio.
8. Explicar o impacto do aumento da *cache* sobre o desempenho e o custo.
9. Descrever como a política de substituição “Least Recently Used” afeta o desempenho.
10. Calcular o custo total de armazenamento (€/GB) de uma hierarquia mista (SSD + HDD + RAM).

Parte C — Analisar

11. Justificar o papel da localidade temporal e espacial no projeto de hierarquias.
12. Comparar MRAM e Flash quanto a durabilidade e velocidade.
13. Analisar como a memória virtual expande a hierarquia lógica.
14. Qual seria o impacto de eliminar a *cache* L2 num processador moderno?
15. Discutir o equilíbrio entre custo e desempenho em arquiteturas híbridas (RAM + SSD).

Questões — Memória Secundária (NotebookLM/ChatGPT)

Verdadeiro-Falso

1. HDD é memória volátil.
2. SSD usa armazenamento baseado em transístores.
3. MRAM é não volátil.
4. SSDs não têm tempo de busca mecânico.
5. Memória cache é mais barata que SSD.
6. Flash NAND é mais rápida que NOR.

Escolha múltipla

7. O tempo médio de acesso depende de:
    - (A) velocidade CPU
    - (B) taxa de acerto

- (C) tipo de barramento
- (D) tamanho do disco

8. Hierarquia correta:
   - (A) CPU→SSD→Cache→RAM→Disco
   - (B) CPU→Cache→RAM→SSD→Disco
   - (C) Cache→CPU→RAM→SSD→Disco
   - (D) CPU→Disco→RAM→SSD→Cache
9. Política LRU substitui:
   - (A) dado mais novo
   - (B) menos usado
   - (C) dado mais antigo
   - (D) nenhuma das anteriores
10. A hierarquia de memória é baseada em:
    - (A) custo/bit
    - (B) largura de banda
    - (C) ambos
    - (D) nenhuma dos anteriores
11. O tempo de acesso médio reduz quando:
    - (A) miss aumenta
    - (B) hit aumenta
    - (C) ambos
    - (D) nenhuma das anteriores

Resposta curta

12. Explicar o conceito de *latência rotacional*.
13. Qual a vantagem do uso de cache de disco?
14. Calcular o tempo médio para hit=0.9 e miss=50 ns, tempo cache=1 ns.
15. Porque é que a localidade temporal aumenta a taxa de acerto?
16. Explicar porque é que a DRAM precisa de refresh.
17. Como é que é medido o desempenho de um SSD?
18. Diferenciar *hit rate* e *miss penalty*.
19. Explicar o equilíbrio entre custo, velocidade e capacidade.

2. Aula Prática Laboratorial (2h)

Objetivos:

- **Implementar** ciclos (for, while, do...while) em Assembly MIPS.
- **Manipular** registos e instruções de salto (beq, bne, blt, bge).
- **Desenvolver** algoritmos iterativos (Fibonacci, conversão binária, mínimos).
- **Reforçar** a compreensão do controlo de fluxo e das condições de paragem.
- **Aplicar** PC-PAA no ChatGPT para compreender passo a passo cada instrução.

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Pequenos exercícios guiados no MARS, desafios de codificação incremental (partindo de exemplos incompletos), e momentos de reflexão em pares, onde os estudantes comparam abordagens e discutem erros.
PC-PAA no ChatGPT	Autoavaliação: o estudante descreve o que aprendeu, testa variantes e pede *feedback* ao ChatGPT
Feedback imediato	Discussão dialogada após cada exercício.
Progressão Bloom	Recordar — revisão de instruções de salto e registos básicos (beq, bne, j, $t0–$t9). Aplicar — tradução de algoritmos simples (loops for, while, do...while) para código MIPS. Analisar — depuração, otimização e compreensão de dependências de controlo e *branch delay slots*.

Folha de Trabalho

Parte A — Recordar

1. Escrever um loop for(i=0; i<10; i++) em Assembly.
2. Traduzir while(a<20) a=a+1; em Assembly.
3. Explicar a diferença entre bne e beq.

Parte B — Aplicar

4. Programa que imprime um número em binário (32 bits).
5. Adaptar o programa anterior para imprimir espaços a cada 4 bits.
6. Modificar para suprimir zeros à esquerda.
7. Escrever um programa para gerar os 10 primeiros termos de Fibonacci.
8. Adaptar para o utilizador introduzir o número de termos.

Parte C — Analisar

9. Implementar um programa que lê o nome e a idade e imprime o mais novo (termina com *idade = 0*).
10. Construir um programa que desenhe barras de temperatura com asteriscos (*).
11. Criar simulação de passeio aleatório (movimento +1 ou -1 até ±d).

Questões — Assembly (NotebookLM/ChatGPT)

Verdadeiro-Falso

1. bne salta quando registos são iguais.
2. j é salto condicional.
3. bgez verifica se o registo é ≥0.
4. j guarda endereço de retorno em $ra.
5. O loop do...while executa sempre pelo menos uma vez.
6. jr $ra faz retorno de função.

7. nop evita dependência de dados.

Escolha múltipla

8. Instrução para saltar se menor que zero:
   - (A) bgtz
   - (B) bltz
   - (C) blez
   - (D) bgez
9. Qual instrução para testar igualdade?
   - (A) beq
   - (B) bne
   - (C) slt
   - (D) nenhuma das anteriores
10. slt $t0,$t1,$t2 significa:
    - (A) $t0=1 se t1<t2
    - (B) $t0=1 se t1>t2
    - (C) $t0=1 se t1=t2
    - (D) nenhuma das anteriores
11. A instrução blez salta quando:
    - (A) >0
    - (B) ≤0
    - (C) <0
    - (D) nenhuma das anteriores

Resposta curta

12. Escrever for(i=0;i<5;i++) em MIPS.
13. Explicar a diferença entre do...while e while.
14. Como evitar *branch delay slot* incorreto?
15. Qual o papel de $ra em funções?
16. Traduzir: “Enquanto x≤10, soma+=x”.
17. Explicar a estrutura de um ciclo for em Assembly.
18. Porque é que slt é útil em comparações compostas?
19. Escrever código para imprimir 5 termos da sequência Fibonacci.
20. Explicar porque é que o controlo de fluxo é fundamental na execução de programas.

Exercícios

1. Escrever um programa em MIPS que: (i) Pede ao utilizador um inteiro N. (ii) Calcula a soma de todos os inteiros de 1 até N (isto é, 1+2+...+N). (iii) Imprime o resultado. Se N <= 0, o programa deve imprimir 0 como soma.
2. Escrever um programa que: Lê inteiros ao utilizador repetidamente. Termina quando o utilizador introduzir 9999 (sentinela). No fim, imprime: (i) quantos valores positivos foram introduzidos; (ii) quantos negativos; (iii) quantos zeros.
3. Escrever um programa que: Pede ao utilizador dois inteiros A e B com A <= B. Para cada número n de A até B, imprime: n: seguido da representação binária de 32 bits desse número. Termina com uma linha em branco.

4. Escrever um programa que: Lê pelo menos uma nota inteira entre 0 e 20. Continua a ler notas enquanto o utilizador responder 1 à pergunta “Continuar? (1=sim, 0=nao)”. No final, imprime a média aritmética das notas lidas (em inteiro).
5. Escrever um programa que: Lê inteiros do utilizador até ser introduzido 0 (0 não conta). No final, imprime: (i) o mínimo, (ii) o máximo, (iii) a média inteira dos valores lidos. Se o utilizador introduzir 0 logo à primeira, o programa deve indicar que não foram introduzidos valores.

Como aplicar o PC-PAA (autoestudo).

- Para os exercícios (1-5), escrever a solução e pedir ao ChatGPT para gerar uma solução. Aplicar o PC-PAA da forma descrita para se observar as diferenças.
- Exercício 1
  - Pedir ao ChatGPT: “*Ajuda-me a resolver o exercício do somatório com while.*”
  - O ChatGPT questiona: “*Quais são as variáveis e condições necessárias para o loop?*”, “*Em que momento o contador deve parar?*”
  - O estudante descreve o plano lógico; o ChatGPT valida.
  - Depois, o estudante escreve o código MIPS.
  - O ChatGPT revê:
    - se a condição de paragem (bgt) está correta,
    - se o incremento é no fim do ciclo,
    - se a soma é feita no local certo.
  - Por fim, o ChatGPT propõe: “*Agora adapta o programa para somar apenas números pares.*”
- Exercício 2
  - O estudante escreve o algoritmo em pseudocódigo.
  - O ChatGPT pergunta: “Quantos casos de decisão existem?” e “Como traduzes isso para MIPS?”
  - Escrever o código e testar.
  - A IA analisa a sequência dos *branches*, deteta possíveis erros de salto (“loop infinito”) e propõe otimizações.
  - ChatGPT lança um desafio: “Consegues reescrever o programa usando apenas instruções slt e bne sem bltz?”
- Exercício 3
  - ChatGPT pede: “Explica como separarias este problema em dois ciclos e o que faz cada um.”
  - Desenhar o diagrama de fluxo (pedir ao ChatGPT para gerar).
  - Depois, escrever o código; a IA verifica: se o contador de bits é reiniciado no local correto; se a condição bgt/bltz é coerente.
  - O ChatGPT propõe um passo seguinte: “Adapta o programa para imprimir apenas os últimos 8 bits de cada número.”
- Exercício 4

- ChatGPT pergunta: “Porque é que do…while é mais adequado do que while neste caso?”
- Escrever o código e o tutor: (i) valida a ordem das instruções (corpo antes da condição), (ii) verifica se a condição de repetição (beq $t1,1) está correta.
- A IA simula testes: “O que acontece se o utilizador introduzir 0 notas? O programa comporta-se bem?”

- Exercício 5
  - ChatGPT pede: “Como inicializas as variáveis min e max antes da primeira leitura?”
  - Responder e implementar.
  - O tutor analisa passo a passo: (i)se a flag de primeiro elemento (t7) foi usada corretamente, (ii) se a atualização de min/max ocorre apenas quando necessário.
  - A IA lança o desafio: “E se quisesses armazenar também a posição (índice) do mínimo e do máximo?”

Notas finais de estudo autónomo com o PC-PAA

1. **Alternância**: tentar prever resultados (papel) → validar no MARS → pedir ao ChatGPT para explicar as diferenças.
2. **Socraticamente**: pedir ao ChatGPT para não dar o código final de imediato, mas solicita pistas graduais.
3. **Reflexão**: no final, solicitar um resumo das dificuldades e um plano de treino (3 exercícios novos criados pelo ChatGPT, baseados nos erros).

BloomArch Learning Framework

11ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**
- **20 questões por tópico (40 no total)**
- **5 exercícios laboratoriais com soluções detalhadas**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- **Compreender** o funcionamento dos sistemas de Entrada e Saída (I/O) e a interação entre CPU, controladores e dispositivos periféricos.
- **Distinguir** os três mecanismos de transferência de dados (*polling*, interrupções e DMA) avaliando as suas implicações no desempenho do sistema.
- **Identificar** os componentes e etapas fundamentais de uma operação de I/O, desde o pedido até à conclusão da transferência.
- **Calcular** o tempo total e a eficiência de transferência em sistemas com diferentes métodos de I/O, relacionando com a ocupação da CPU.
- **Analisar** o impacto das diferentes estratégias de I/O (*polling*, interrupções, DMA) no desempenho global do sistema, justificando escolhas de projeto em cenários concretos.

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Os estudantes constroem o conhecimento através de explicações curtas seguidas de pequenos desafios práticos, discutidos em grupo. Esta abordagem promove a participação, o raciocínio crítico e a ligação entre teoria e prática.
PC-PAI no ChatGPT	O Prompt Completo – Plano de Autoestudo Interativo (PC-PAI) é usado após a aula para revisão e autoavaliação guiada. O ChatGPT conduz um diálogo estruturado de perguntas–respostas–feedback, permitindo consolidar conceitos e praticar cálculos de desempenho em I/O.
Feedback imediato	O *feedback* ocorre durante e após a aula, tanto pelo docente como pelo ChatGPT. É formativo e explicativo, ajudando o estudante a corrigir erros e compreender o raciocínio subjacente, reforçando a aprendizagem de forma imediata.
Progressão Bloom	As atividades seguem a lógica de progressão cognitiva: Recordar → Compreender → Aplicar → Analisar. Cada fase aumenta o nível de exigência intelectual, conduzindo o estudante de uma aprendizagem factual para uma compreensão crítica e aplicada do sistema de Entrada e Saída.

Tempo	Atividade	Objetivo de Aprendizagem	Nível Bloom
0–10 min	Introdução e contextualização do tema (exposição dialogada)	Compreender o papel dos sistemas de Entrada e Saída e sua relação com CPU e memória.	Compreender
10–25 min	Explicação dos mecanismos de I/O (polling, interrupções, DMA) com recurso a esquemas e exemplos	Distinguir os três tipos de transferência de dados e as suas implicações no desempenho.	Distinguir
25–40 min	Atividade em pares: identificar componentes de um ciclo de I/O a partir de um diagrama incompleto	Identificar os componentes e etapas fundamentais de uma operação de I/O.	Identificar
40–55 min	Mini exercício guiado: cálculo do tempo médio de I/O via *polling vs* interrupções	Calcular o tempo total e a eficiência de transferência em diferentes métodos.	Calcular
55–70 min	Discussão orientada: comparação entre eficiência de DMA e interrupções	Analisar o impacto das estratégias de I/O no desempenho global do sistema.	Analisar
70–90 min	Resolução coletiva de um caso prático (exemplo: comunicação entre CPU e disco via DMA)	Aplicar os conceitos de controlo de I/O num cenário prático.	Aplicar/Analisar
90–105 min	Revisão sumativa: questões rápidas (V/F e múltipla escolha) + debate	Consolidar a compreensão e corrigir conceções erradas.	Compreender / Aplicar
105–120 min	Síntese final e preparação para	Integrar conhecimentos e preparar o autoestudo interativo.	Analisar / Autoavaliar

	o PC-PAI (ChatGPT)		

Folha de Trabalho

Parte A — Recordar

1. Definir I/O mapeado em memória e I/O isolado.
2. Identificar as etapas de uma transferência de I/O por *polling*.
3. Descrever o papel do registo de estado no controlador de I/O.
4. Explicar o que significa barramento partilhado.
5. Quais os três tipos principais de I/O discutidos no livro adotado.

Parte B — Aplicar

6. Calcular o tempo total de I/O via *polling*, sabendo que cada ciclo ocupa 200 µs e o dispositivo requer 10 ciclos.
7. Mostrar como uma interrupção reduz a ocupação da CPU em relação ao *polling*.
8. Escrever um pseudocódigo que simule I/O por interrupções.
9. Explicar como o DMA transfere dados sem intervenção da CPU.
10. Identificar dois exemplos práticos de dispositivos que utilizam DMA.

Parte C — Analisar

11. Comparar *polling* e interrupções em termos de latência e eficiência.
12. Explicar por que motivo o DMA necessita de arbitragem de barramento.
13. Analisar o impacto do tamanho do buffer no desempenho de I/O.
14. Discutir as vantagens do I/O mapeado em memória sobre o I/O isolado.
15. Avaliar o papel da hierarquia de barramentos na redução de congestionamento.

Questões — Memória Secundária (NotebookLM/ChatGPT)

Verdadeiro-Falso

1. No I/O mapeado em memória, as instruções de leitura e escrita são as mesmas usadas para RAM.
2. O *polling* é mais eficiente que as interrupções em sistemas multitarefa.
3. O DMA permite à CPU realizar outras operações durante a transferência de dados.
4. O barramento de dados transporta endereços de dispositivos.
5. Um buffer de maior capacidade reduz a frequência de interrupções.

Escolha múltipla

6. O principal problema do polling é:
    - (A) Complexidade do hardware
    - (B) Ocupação constante da CPU
    - (C) Falta de sincronização
    - (D) Impossibilidade de acesso direto

7. O sinal de interrupção é gerado por:
    - (A) CPU
    - (B) Controlador de memória
    - (C) Dispositivo de I/O
    - (D) DMA
8. O DMA necessita de arbitragem de barramento para:
    - (A) Resolver empates com a cache
    - (B) Partilhar o barramento com a CPU
    - (C) Controlar interrupções
    - (D) Reduzir a latência
9. A vantagem do I/O mapeado em memória é:
    - (A) Mais segurança
    - (B) Menor custo
    - (C) Simplicidade de instruções
    - (D) Requer menos barramentos
10. A eficiência do I/O depende fortemente de:
    - (A) Largura de banda do barramento
    - (B) Número de dispositivos
    - (C) Tipo de CPU
    - (D) Tamanho da cache

Resposta curta

11. Descrever a sequência básica de uma interrupção de I/O.
12. O que é latência de serviço de dispositivo?
13. Explicar o conceito de controlador de I/O.
14. Como o Sistema Operativo comunica com o controlador de dispositivos?
15. Dar um exemplo de dispositivo que usa interrupções em tempo real.

Problemas

16. Calcule o ganho de eficiência ao passar de polling (2 ms) para interrupção (0,5 ms).
17. Num sistema com DMA, se a transferência ocupa 1 µs e o CPU demora 10 µs por instrução, qual a melhoria percentual?
18. Num sistema com buffer duplo, quantos acessos de CPU são evitados a cada 1000 bytes transferidos?
19. Analisar o impacto de interrupções simultâneas em sistemas de prioridade.
20. Explicar como o DMA pode gerar overlapping de I/O e processamento.

2. Aula Prática Laboratorial (2h)

Objetivos:

- **Implementar** rotinas de entrada e saída em Assembly MIPS, utilizando *syscalls* para leitura e escrita de dados (inteiros, *strings* e caracteres).

- **Manipular** registos e argumentos de sistema ($v0, $a0, $a1, $t0, etc.) de forma correta, para controlar operações de I/O e compreender o fluxo de dados entre CPU e dispositivos simulados
- **Desenvolver** pequenos programas interativos em Assembly, que combinem operações de leitura, processamento e escrita de resultados no ecrã, consolidando a lógica sequencial e condicional.
- **Reforçar** a ligação entre teoria e prática da arquitetura de Entrada e Saída, reconhecendo o papel das *syscalls* como abstrações de interrupções de I/O e do sistema operativo.
- **Aplicar** o Prompt Completo – Plano de Autoestudo Assembly (PC-PAA) no ChatGPT para testar, depurar e otimizar programas MIPS, com base no feedback automático e explicações detalhadas.

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Os estudantes programam e testam rotinas de I/O em Assembly no simulador MARS, resolvendo pequenos desafios guiados pelo docente. A prática e a discussão entre pares promovem compreensão e autonomia.
PC-PAA no ChatGPT	Autoavaliação: o estudante descreve o que aprendeu, testa variantes e pede *feedback* ao ChatGPT
Feedback imediato	Durante a sessão, o docente dá feedback formativo sobre estrutura, lógica e uso de registos. O ChatGPT fornece feedback complementar no pós-aula, explicando erros e sugerindo melhorias.
Progressão Bloom	As atividades seguem a sequência Recordar → Compreender → Aplicar → Analisar → Criar, permitindo evoluir da reprodução de código até à análise e desenvolvimento de programas originais de Entrada e Saída.

Folha de Trabalho

Parte A — Recordar

1. Qual a *syscall* para imprimir inteiro?
2. Qual o código para ler um carácter?
3. Que registo transporta o código de *syscall*?
4. O que acontece após *syscall* ser executado?
5. Qual a diferença entre leitura de *string* e de inteiro?

Parte B — Aplicar

6. Programa que lê um inteiro e imprime o dobro.
7. Programa que lê dois inteiros e imprime a soma.
8. Programa que lê uma *string* e conta o número de caracteres.
9. Programa que lê 5 números e imprime a média.
10. Programa que imprime “Olá Mundo” seguido do número de letras.

Parte C — Analisar

11. Programa que lê nomes até encontrar “fim” e imprime todos.
12. Programa que grava uma sequência numérica num ficheiro (*syscall* 13/15).
13. Programa que lê do ficheiro e imprime no ecrã.
14. Programa que soma bytes lidos até EOF.
15. Programa que mostra como combinar *syscalls* 4, 5, 8 e 10 para I/O composto.

Questões — Assembly (NotebookLM/ChatGPT)

Verdadeiro-Falso

1. O registo $v0 guarda o código da syscall.
2. Syscalls bloqueiam sempre a CPU.
3. O registo $a0 contém o endereço da string para syscall 4.
4. syscall 10 encerra o programa.
5. O MARS usa interrupções reais para syscalls.

Escolha múltipla

6. Para ler um inteiro:
   - (A) syscall 1
   - (B) syscall 4
   - (C) syscall 5
   - (D) syscall 8
7. Para imprimir um carácter:
   - (A)10
   - (B) 11
   - (C) 4
   - (D) 13
8. Para abrir ficheiro:
   - (A) 15
   - (B) 13
   - (C) 14
   - (D) 17
9. O código 14 é usado para:
   - (A) Escrever ficheiro
   - (B) Fechar ficheiro
   - (C) Ler ficheiro
   - (D) Abrir ficheiro

Resposta curta

10. Qual é a função de $v0 em *syscalls*?
11. Como são passados os parâmetros a uma *syscall*?
12. O que acontece se $v0 contiver um código inválido?
13. Porque é que *syscall* simula interrupções?
14. Qual é a diferença entre syscalls 8 e 13?

Problemas

15. Programa que lê 10 inteiros e escreve num ficheiro.
16. Programa que copia o conteúdo de um ficheiro para outro.
17. Programa que converte letras minúsculas em maiúsculas.
18. Programa que imprime “erro” se o ficheiro não for encontrado.
19. Programa que mede o tempo de execução entre duas syscalls.

Exercícios

1. Escrever um programa em Assembly MIPS que: Lê um inteiro N do utilizador. Imprime todos os números inteiros de 1 até N, um por linha. Se N <= 0, não deve imprimir nada.
2. Escrever um programa em MIPS que: Lê um inteiro N. Calcula a soma de todos os números pares de 1 até N e imprime o resultado.
3. Fazer um programa que: Lê inteiros do utilizador repetidamente. Termina quando o utilizador introduzir 9999. No fim, imprime quantos números foram: positivos, negativos, iguais a zero.
4. Escrever um programa que: Lê um inteiro N > 0. Lê N inteiros. Calcula o mínimo, o máximo e a média inteira desses valores. Imprime os três resultados.
5. Implementa um programa com um menu interativo:
   O programa apresenta repetidamente o seguinte menu:
   1 - Somar dois inteiros
   2 - Contar de 1 ate N
   3 - Sair
   Opcao:
   Se o utilizador escolher:
   1: pede dois inteiros e imprime a soma.
   2: pede N e imprime de 1 até N.
   3: termina o programa.
   Qualquer outra opção → imprime “Opcao invalida” e volta ao menu.

Como aplicar o PC-PAA (autoestudo).

- Para os exercícios (1-5), escrever a solução e pedir ao ChatGPT para gerar uma solução. Aplicar o PC-PAA da forma descrita para se observar as diferenças.
- Exercício 1
  - No ChatGPT, o estudante pode escrever algo como: “Estou a treinar ciclos em MIPS. Eis o meu programa para imprimir de 1 a N. Analisa o meu código, explica se a condição do ciclo está correta, se a inicialização está bem feita e sugere melhorias.”
  - O PC-PAA pode:
    - Perguntar: “O que acontece se N for 0 ou negativo?”
    - Sugerir: “Consegues adaptar o programa para imprimir só os pares?”

        - Explicar por que bgt $t0, $s0, END é equivalente a while (i <= N).
- Exercício 2
    - O estudante pode pedir ao ChatGPT: “Quero que me ajudes a entender o meu programa que soma números pares até N em MIPS. Verifica se a condição para par está correta, se a soma é feita no sítio certo e faz-me perguntas de compreensão.”
    - O PC-PAA pode:
        - Perguntar: “O que faz exatamente andi $t2, $t0, 1?”
        - Pedir ao estudante para explicar: “Porque usaste bne $t2, $zero, SKIP em vez de beq?”
        - Propor variação: “Agora soma apenas os ímpares.”
- Exercício 3
    - O estudante pode dizer ao ChatGPT: “Este é o meu programa MIPS que conta positivos, negativos e zeros até inserir 9999. Faz uma análise passo a passo do fluxo de controlo, indica possíveis erros lógicos e sugere como simplificar o código.”
    - O PC-PAA pode:
        - Pedir ao estudante que desenhe (ou explique verbalmente) um diagrama de fluxo do programa.
        - Perguntar: “O que acontece se trocares a ordem dos testes (bltz antes do beq $t0,$zero)?”
        - Sugerir: “Consegues reescrever com menos saltos?”
- Exercício 4
    - O estudante pode usar o ChatGPT assim: “Este é o meu programa MIPS para calcular min, max e média de N valores. Verifica se a lógica de inicialização do primeiro valor está correta, se a atualização de min/max está bem implementada e ajuda-me a corrigir o cálculo da média passo a passo.”
    - O PC-PAA pode:
        - Questionar: “Por que precisas de uma flag para o primeiro valor?”
        - Fazer o estudante descobrir sozinho a necessidade de guardar N original.
        - Sugerir uma extensão: “Agora adiciona também o desvio padrão (em inteiro, aproximado).”
- Exercício 5
    - O estudante pode dizer ao ChatGPT: “Desenvolvi este programa de menu em MIPS. Quero que atues como tutor: verifica se o fluxo de controlo está correto, se há casos em que o programa entra em loop infinito, e ajuda-me a tornar o menu mais robusto (por exemplo, limpar a entrada ou adicionar mais opções).”
    - O PC-PAA pode:
        - Perguntar: “O que aconteceria se o utilizador introduzisse texto em vez de um número?”
        - Propor: “Adiciona uma opção 4 que volta a executar a última operação.”

- Ajudar o estudante a reorganizar o código em “subrotinas” (funções) para cada opção, introduzindo jal e $ra.

Notas finais de estudo autónomo com o PC-PAA

1. **Alternância**: tentar prever resultados (papel) → validar no MARS → pedir ao ChatGPT para explicar as diferenças.
2. **Socraticamente**: pedir ao ChatGPT para não dar o código final de imediato, mas solicita pistas graduais.
3. **Reflexão**: no final, solicitar um resumo das dificuldades e um plano de treino (3 exercícios novos criados pelo ChatGPT, baseados nos erros).

BloomArch Learning Framework

12ª semana
Versão 1.0

Conteúdo

- **Plano TP (2h) e Plano Lab (2h) com BALF**
- **Folhas de trabalho**
- **20 questões por tópico (40 no total)**
- **5 exercícios laboratoriais com soluções detalhadas**

1. Aula Teórico-Prática (2h)

Objetivos de aprendizagem

- **Compreender:** O ciclo de instrução (*fetch–decode–execute*).
- **Identificar:** Os componentes internos do processador e os seus papéis (ALU, UC, registradores).
- **Distinguir:** Entre fluxo de dados e fluxo de controlo.
- **Calcular:** Tempos de execução e latências em ciclos de instrução.
- **Analisar:** O impacto do *pipelining* e das dependências de dados no desempenho.

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Explicação intercalada com microproblemas (ex.: identificar o caminho de dados de uma instrução Load).
PC-PAI no ChatGPT	O Prompt Completo – Plano de Autoestudo Interativo (PC-PAI) é usado após a aula para revisão e autoavaliação guiada. O ChatGPT conduz um diálogo estruturado de perguntas–respostas–feedback, permitindo consolidar conceitos e praticar cálculos de Tempos de execução e latências em ciclos de instrução.
Feedback imediato	O *feedback* ocorre durante e após a aula, tanto pelo docente como pelo ChatGPT. É formativo e explicativo, ajudando o estudante a corrigir erros e compreender o raciocínio subjacente, reforçando a aprendizagem de forma imediata.
Progressão Bloom	As atividades seguem a lógica de progressão cognitiva: Recordar → Compreender → Aplicar → Analisar. Cada fase aumenta o nível de exigência intelectual, conduzindo o estudante de uma aprendizagem factual para uma compreensão crítica e aplicada da Estrutura do Processador e Funções.

Tempo	Atividade	Objetivo de Aprendizagem	Nível Bloom
0–10 min	Revisão guiada dos conceitos-chave: CPU,	Compreender o papel de cada componente interno do processador	Compreender

	UC, ALU, registos, ciclo de instrução		
10–20 min	Microproblema em pares: “Identifica o caminho percorrido por uma instrução LOAD”	Identificar o percurso de dados e os registos envolvidos	Recordar → Compreender
20–35 min	Explicação dialogada: fluxo de controlo vs. fluxo de dados	Distinguir entre sinais de controlo e fluxo de dados num processador	Compreender
35–50 min	Exercício orientado: cálculo do número de ciclos numa sequência de instruções com e sem pipeline	Calcular tempo de execução com diferentes modelos (single-cycle vs. pipeline)	Aplicar
50–65 min	Discussão guiada: hazards (RAW, WAR, WAW), stalls, forwarding	Analisar o impacto das dependências no desempenho	Analisar
65–80 min	PC-PAI no ChatGPT → Questionário interativo (fetch–decode–execute, UC, ALU, registos, pipeline)	Consolidar a compreensão e aplicar conceitos em exercícios de diálogo	Aplicar → Analisar
80–90 min	Feedback imediato + síntese final: o que causa stalls? como o branch prediction melhora desempenho?	Avaliar raciocínios e corrigir conceções erradas	Analisar / Avaliar
90–120 min	(continuação se necessário + dúvidas)	—	—

Folha de Trabalho

Parte A — Recordar

1. Descreve as principais fases do ciclo de instrução.
2. Define a função da unidade de controlo (UC).
3. O que é um registrador e que tipos existem?
4. Qual o papel da ALU no processador?
5. O que representa o PC (Program Counter)?

Parte B — Aplicar

6. Explica o percurso dos dados numa instrução do tipo *Load*.
7. Mostra como ocorre a execução de uma instrução ADD em termos de sinais de controlo.
8. Determina o número de ciclos necessários para executar 10 instruções sem e com pipeline (CPI=1).

9. Calcula o tempo total de execução se o clock for 2 GHz.
10. Identifica dependências de dados entre instruções consecutivas num trecho MIPS.

Parte C — Analisar

11. Analisa o impacto das dependências RAW (Read After Write) no desempenho.
12. Explica a função do *branch predictor*.
13. Diferencia pipeline estrutural, de dados e de controlo.
14. Avalia o impacto de *hazards* na eficiência do pipeline.
15. Discute o equilíbrio entre complexidade de hardware e desempenho.

Questões — Memória Secundária (NotebookLM/ChatGPT)

Verdadeiro-Falso

1. A UC coordena o fluxo de dados no processador.
2. O PC armazena resultados de operações aritméticas.
3. A ALU é responsável pelo ciclo de busca de instruções.
4. A execução de instruções é sempre sequencial.
5. O pipeline elimina completamente as dependências de dados.

Escolha múltipla

6. A função do registrador IR é:
   - (A) Guardar resultados lógicos
   - (B) Conter a instrução atual
   - (C) Controlar interrupções
   - (D) Calcular endereços
7. A latência do pipeline depende de:
   - (A) Número de fases
   - (B) Clock
   - (C) Ambos
   - (D) Nenhum
8. O branch predictor serve para:
   - (A) Reduzir ciclos de clock
   - (B) Antecipar o destino de saltos
   - (C) Eliminar stalls
   - (D) Controlar exceções
9. O caminho de dados inclui:
   - (A) UC e ALU
   - (B) ALU, registradores e memória
   - (C) Apenas memória
   - (D) UC e *cache*
10. O hazard de controlo ocorre devido a:
    - (A) Interrupções
    - (B) Dependências entre saltos
    - (C) Falta de recursos
    - (D) *Caches* inconsistentes

Resposta curta

11. Explicar a diferença entre UC cablada e microprogramada.
12. O que é forwarding no pipeline?
13. Porque se usa pipeline?
14. Qual a diferença entre throughput e latência?
15. Que estratégias mitigam hazards?

Problemas

16. Calcular o CPI efetivo se 30% das instruções sofrem hazard de 2 ciclos.
17. Num processador com 5 estágios, qual o speedup teórico para 100 instruções?
18. Estimar o tempo total se o clock = 1 ns e 5% de stalls.
19. Analisar o impacto de duplicar a frequência no throughput.
20. Desenha o fluxo de execução para uma instrução LW.

2. Aula Prática Laboratorial (2h)

Objetivos:

- **Implementar:** Ciclos e condições em Assembly.
- **Manipular:** Registos de controlo e dados.
- **Desenvolver:** Programas que simulem operações internas de CPU.
- **Reforçar:** O entendimento do fluxo *fetch–execute*.
- **Aplicar** o Prompt Completo – Plano de Autoestudo Assembly (PC-PAA) no ChatGPT para testar, depurar e otimizar programas MIPS, com base no feedback automático e explicações detalhadas.

Estratégia pedagógica (BALF)

Etapa	Descrição
Aprendizagem ativa	Pequenas tarefas práticas de execução e depuração no MARS.
PC-PAA no ChatGPT	Autoavaliação: o estudante descreve o que aprendeu, testa variantes e pede *feedback* ao ChatGPT
Feedback imediato	Correção e explicação passo a passo dos fluxos de instrução.
Progressão Bloom	As atividades seguem a sequência Recordar → Compreender → Aplicar → Analisar → Criar, permitindo evoluir da reprodução de código até à análise e desenvolvimento de pequenos programas .

Folha de Trabalho

Parte A — Recordar

1. Qual o papel de $ra e jal nas chamadas de função?
2. O que faz a instrução jr $ra?
3. Como são passados argumentos para funções?
4. O que é uma pilha (*stack*)?

5. Como se guarda o contexto antes de uma chamada?

Parte B — Aplicar

6. Implementar uma função que calcula x^2.
7. Criar função soma(a,b) e imprimir resultado.
8. Adicionar função recursiva para fatorial.
9. Criar rotina que devolve o maior de três números.
10. Simular ciclo *fetch–decode–execute* usando três sub-rotinas.

Parte C — Analisar

11. Reescrever função recursiva para versão iterativa.
12. Comparar desempenho entre versões recursiva e iterativa.
13. Explicar o impacto do *stack overflow*.
14. Identificar dependências nos registos $a0–$a3 e $v0.
15. Discutir a diferença entre instruções jal e jalr.

Questões — Assembly (NotebookLM/ChatGPT)

Verdadeiro-Falso

1. A instrução jal guarda o endereço de retorno em $ra.
2. O *stack pointer* aponta sempre para o topo ocupado da pilha.
3. Funções recursivas não utilizam a pilha.
4. $a0 e $v0 são registos equivalentes.
5. jr $ra provoca salto para o início do programa.

Escolha múltipla

6. O registo $ra serve para:
   - (A) Guardar resultados
   - (B) Guardar o endereço de retorno
   - (C) Contar instruções
   - (D) Nenhuma
7. A instrução jalr difere de jal por:
   - (A) Usar endereço variável
   - (B) Guardar retorno em $a0
   - (C) Invocar syscall
   - (D) Não guardar retorno
8. O stack pointer ($sp) é alterado por:
   - (A) addiu
   - (B) sw/lw
   - (C) Ambas
   - (D) Nenhuma
9. A chamada de função típica inclui:
   - **(**A) jal, jr e uso de $ra
   - (B) Apenas jal
   - (C) Apenas jr
   - (D) Nenhuma das anteriores

10. A pilha cresce:
    - (A) Para endereços maiores
    - (B) Para endereços menores
    - (C) Horizontalmente
    - (D) De forma aleatória

Resposta curta

11. Explicar o fluxo jal → execução → jr.
12. Como é que se preserva $ra numa função recursiva?
13. O que acontece se o *stack overflow*?
14. Porque é necessário usar sw/lw antes de chamadas aninhadas?
15. Explica o papel de $sp e $fp na gestão de funções.

Problemas

16. Escreve uma função produto(a,b) sem usar multiplicação.
17. Implementa máximo(a,b) e devolve via $v0.
18. Usa loop para calcular soma de 1 a N.
19. Implementa uma função que imprime os N primeiros pares.
20. Reescreve função recursiva fatorial para versão iterativa.

Exercícios

1. Implementar um programa em MIPS que:
    - Pede ao utilizador um número N (1 ≤ N ≤ 100) de notas.
    - Lê N notas inteiras entre 0 e 20 e guarda-as num *array*.
    - Calcula:
        - Nota mínima
        - Nota máxima
        - Média (inteira)
    - Constrói um histograma das notas (*array hist*[21] onde *hist*[i] é o nº de alunos com nota i).
    - Imprime:
        - Min, Max, Média
        - O histograma (linha "nota: contagem").
2. Escreve um programa que:
    - Lê N inteiros (positivos, negativos e zero) para um *array*.
    - Cria um segundo *array* contendo apenas os valores positivos.
    - Ordena esse segundo *array* por inserção (*Insertion Sort*).
    - Imprime o *array* de positivos ordenado.
3. Construir um programa que simula uma máquina de estados finitos com 3 estados:
    - S0 (inicial)
    - S1
    - S2

    Regras:
    - Lê uma sequência de caracteres ('a' ou 'b') até '\n'.
    - Transições:
        - Em S0: 'a' → S1; 'b' → S0

    - Em S1: ‘a’ → S1; ‘b’ → S2
    - Em S2: ‘a’ → S2; ‘b’ → S0
  - No fim, imprime o estado final (0, 1 ou 2).
4. Criar um “interpretador” de um pequeno conjunto de instruções (*bytecode*):
   - Memória de programa: *array* de pares (*opcode*, operando).
   - *OpCodes*:
     - 1 → ADD acumulador += operando
     - 2 → SUB acumulador -= operando
     - 3 → MUL acumulador *= operando
     - 4 → END (termina execução)

   O programa deve:
   1. Inicializar o acumulador a 0.
   2. Ler sequencialmente o *array* de instruções.
   3. Executar até encontrar *opcode* 4.
   4. Imprimir o valor final do acumulador.
5. Criar um programa de menu com 3 opções, implementadas cada uma numa função separada:
   1. 1 – Calcular fatorial (iterativo).
   2. 2 – Calcular soma de 1 a N.
   3. 3 – Sair.

   Requisitos:
   - Cada opção é uma função chamada com jal.
   - A função do menu chama repetidamente até o utilizador escolher 3.
   - Deve preservar corretamente $ra e $sX se usados.

Como aplicar o PC-PAA (autoestudo).

- Para os exercícios (1-5), escrever a solução e pedir ao ChatGPT para gerar uma solução. Aplicar o PC-PAA da forma descrita para se observar as diferenças.
- Exercício 1
  - No ChatGPT, o estudante pode escrever algo como: “*Este é o meu programa em MIPS para gerir notas (min, max, média, histograma). Analisa o código, verifica se há erros na gestão de índices e na inicialização do histograma. Explica passo a passo o que acontece no loop de leitura e no loop do histograma. Sugere melhorias e faz-me perguntas de controlo.*”
  - O PC-PAA pode:
    - Detectar erros de off-by-one.
    - Pedir ao aluno para explicar a diferença entre sll (shift) e mul.
    - Propor variações: histograma por intervalos (0–9,10–20).
- Exercício 2
  - O estudante pode pedir ao ChatGPT: “Mostro-te o meu insertion sort em MIPS. Quero que verifiques se a lógica de deslocamento está correta, se não há acessos fora do array e que me expliques a diferença

entre while_j e place_key. Depois, pede-me para desenhar o pseudocódigo a partir do Assembly."
    - O PC-PAA pode:
        - Fazer o aluno reconstruir o pseudocódigo
        - Sugerir: "Agora troca para ordenação decrescente."
        - Identificar erros de endereço ao deslocar B[j].
- Exercício 3
    - O estudante pode dizer ao ChatGPT: "Este é o meu simulador de máquina de estados em MIPS. Explica o diagrama de estados correspondente ao código, verifica se as transições estão corretas e desafia-me a adicionar um quarto estado S3 com novas regras."
    - O PC-PAA pode:
        - Pedir ao aluno para desenhar o diagrama de estados.
        - Sugerir refatorar usando uma tabela de transições (array 3x2).
- Exercício 4
    - O estudante pode usar o ChatGPT assim: "Este é o meu interpretador de bytecode em MIPS. Explica como o laço loop_vm imita o ciclo fetch–decode–execute. Pede-me para desenhar o fluxo de uma instrução ADD e para adicionar um novo opcode 5 para dividir o acumulador por 2"
    - O PC-PAA pode:
        - Fazer o estudante mapear cada etiqueta (OP_ADD, etc.) para a microarquitetura.
        - Pedir para modificar o programa prog e prever o resultado antes de correr.
- Exercício 5
    - O estudante pode dizer ao ChatGPT: "Este é o meu programa de menu modular em MIPS. Verifica se o fluxo de chamadas está correto, se o $ra é sempre preservado e se poderia haver problemas ao aninhar chamadas. Sugere como introduzir uma nova opção para calcular o máximo de dois números reutilizando uma função maximo(a,b)."
    - O PC-PAA pode:
        - Pedir ao estudante para desenhar o call graph (diagrama de chamadas).
        - Sugerir introdução de stack frame mais completo ($sp, $fp).

Notas finais de estudo autónomo com o PC-PAA

4. **Alternância**: tentar prever resultados (papel) → validar no MARS → pedir ao ChatGPT para explicar as diferenças.
5. **Socraticamente**: pedir ao ChatGPT para não dar o código final de imediato, mas solicita pistas graduais.
6. **Reflexão**: no final, solicitar um resumo das dificuldades e um plano de treino (3 exercícios novos criados pelo ChatGPT, baseados nos erros).