

# Arquitetura, programação e aplicações frente a micro-controladores e processadores digitais

Um sistema de computação, ou sistema digital, é um conjunto de vários componentes que coordenam os dispositivos de entrada e saída (I/O), além da memória para a realização do processamento de algum dado. Um sistema de computação pode ser entendido como a união do Hardware (processador principal, memória, dispositivos de I/O e barramentos) e do Software (programa que usa os componentes do hardware). O enfoque deste texto é a descrição dos sistemas digitais: microcontroladores e processadores digitais de sinais (DSP - do inglês *Digital Signal Processor*).

Como é mostrado na Figura 1, os sistemas digitais podem ser agrupados em três principais famílias:

- Lógica padrão: engloba os componentes digitais mais básicos (portas lógicas, flip-flops, decodificadores) que estão disponíveis como circuitos integrados, alguns mais de 30 anos. Eles já vêm sendo usados há anos para projetar sistemas digitais complexos. Eles possuem diversos encapsulamentos devido a diversidade de tipos existentes. O processo de manipulação é complexo e envolve um alto custo. O hardware já vem pronto de fábrica com uma função específica. A ULSI é um exemplo de empresa que lida com este tipo de projeto.
- Circuitos integrados de aplicação específica (ASICs - Application Specific Integrated Circuits): É a solução moderna em termos de hardware para os sistemas digitais. São circuitos integrados projetados para implementar uma aplicação específica. São sistemas mais complexos, que demandam muito tempo de desenvolvimento e muitos recursos. Isso os torna sistemas mais caros, em contrapartida, o sistema resultante oferece um ótimo desempenho com um baixo consumo de energia. Exemplos são os PLDs (dispositivos lógicos programáveis), como o FPGA e até os PSoCs.
- Microcontroladores/Microprocessadores/processadores digitais de sinais (DSP - do inglês *Digital Signal Processor*): Diferente do projeto de sistemas digitais com componentes digitais básicos, esses dispositivos possuem vários tipos de blocos funcionais dentro de uma placa. Eles possuem um hardware fixo, mas com a flexibilidade de adequar o código a cada problema. O grande problema associado a esse tipo de dispositivo é a velocidade, uma vez que a solução de hardware para o seu projeto de sistema digital sempre será mais rápido do que uma solução de software.

Tanto os microcontroladores, microprocessadores e os DSP são circuitos integrados, os quais possuem um núcleo de processamento, memória e periféricos de I/O. Contudo eles possuem algumas diferenças em sua essência. A diferença entre microcontroladores (MC) e

microprocessadores (MP) está no ponto da complexidade envolvida ao longo do processador principal. O MP contém o processador e um conjunto limitado de memória e I/O, além de não ter memória RAM (Random Access Memory), ROM (Read-Only Memory) e outros periféricos no mesmo chip. É necessário incluir componentes externos para torná-lo funcional. Já o MC possui a maioria da memória do sistema e de I/O integrada no chip, incluindo um conjunto fixo de memória RAM, ROM e outros periféricos todos fixos a um chip. As funções dos MP incluem tarefas que necessitam de grandes quantidades de RAM, ROM, portas I/O e que são mais genéricas como o desenvolvimento de software, websites, editor de fotos. Nesse caso não há uma relação forte entre a entrada e saída. No caso dos MC, seu planejamento já é pensando em tasks específicas, onde a relação entre entrada e saída é bem definida. A quantidade de recursos de RAM, ROM e portas I/O são mais limitados e podem todos serem incorporados a um único chip, o que faz com que o tamanho e custo sejam reduzidos. Entre as principais tasks inclui-se: mouse, teclado, máquina de lavar, pendrive, carros, microondas, smartphones, entre outros.

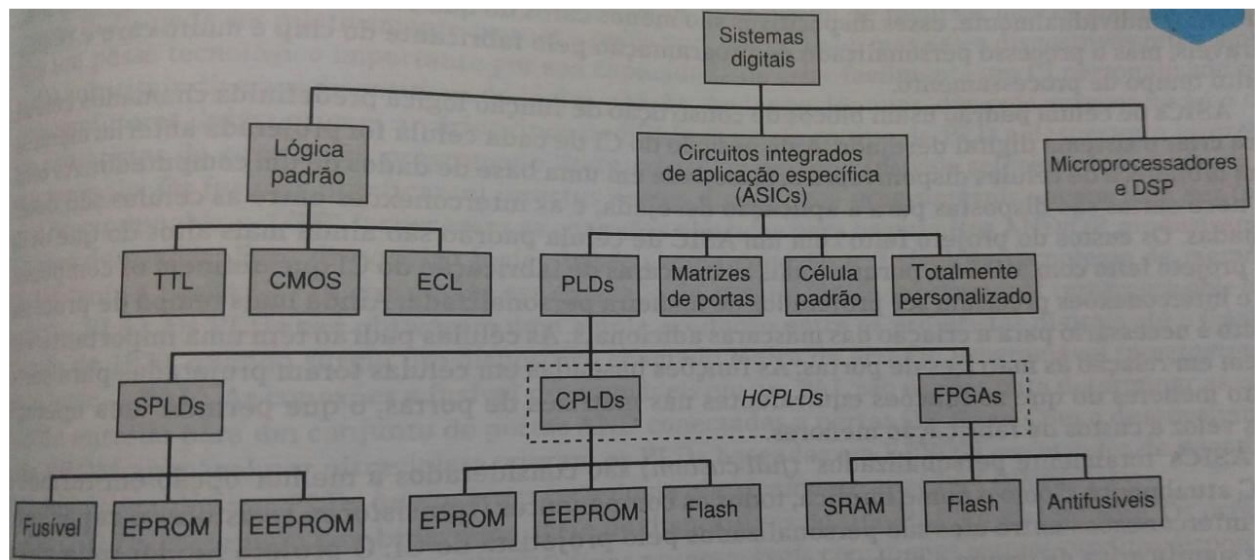


Figura 1: Famílias de sistemas digitais

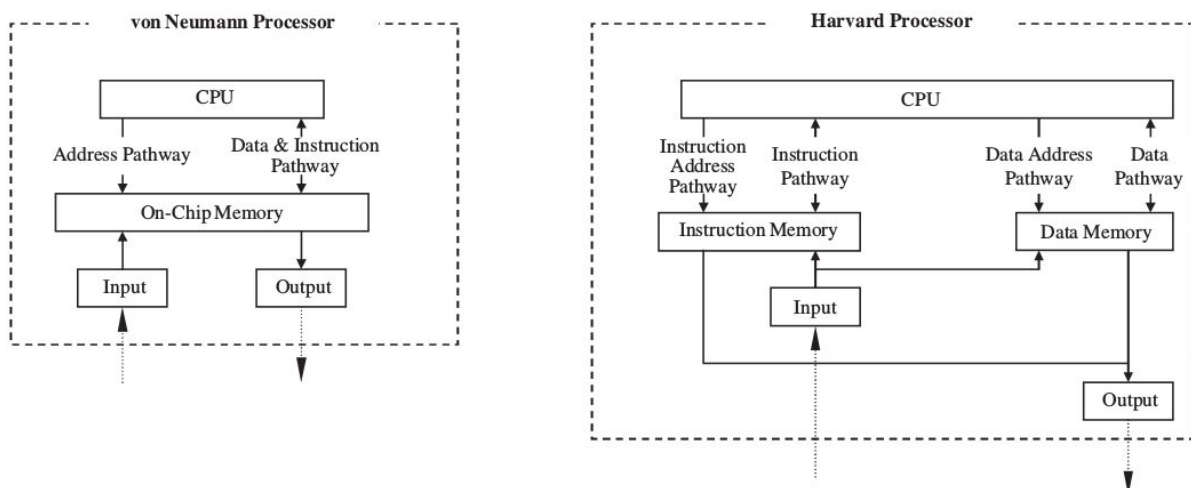
A principal diferença entre MP e DSP é que o segundo é um processador digital especializado em um tipo de sinal específico, como de áudio, vídeo, entre outros e o primeiro é de uso geral. Esse processamento pode ocorrer tanto em tempo real ou off-line. Já a diferença entre os MC e DSP, é baseada na diferença entre MP e MC e que o DSP é ainda mais especializado do que o MP. O foco da utilização dos DSPs são em processamento de áudio e vídeo em sua maioria, mas que envolvem o processamento de um sinal analógico, que será parseado. Além disso, os MCs possuem seu foco de utilização em automação e controle de produtos e periféricos, uma vez que possuem um tamanho, custo e consumo de energia reduzidos quando comparados à forma mais comum de uso dos microprocessadores convencionais.

Os DSPs são de uso mais específicos e já possuem em seu interior toda a estrutura necessária para processar um sinal na forma digital, com entradas e saídas analógicas.

Levando isso em conta, os DSPs possuem circuitos conversores analógico-digital e normalmente são projetados pensando nas operações mais habituais no processamento de um sinal: adição, multiplicação e transferência de dados na memória. Eles são otimizados para realizarem esse tipo de processamento. Instruções de repetição que antecedem instruções numéricas são implementadas a fim de tornar possível a execução de várias instruções em apenas um ciclo de memória, uma vez que a instrução de retorno é descartada para que se permaneça dentro do laço de repetição, o que faz com que o processamento seja mais rápido.

Dentre os diversos elementos presentes no hardware de sistemas computacionais, incluindo MC, MP e DSP (dadas as proporções), destacam-se quatro: processadores, sistema de memória, sistema de entrada e saída (I/O) e barramentos. Cada um possui uma função. O processador/CPU (Central Process Unit) processa instruções, uma a uma, em uma ordem pré-estabelecida. A CPU é a responsável por executar e gerenciar as instruções, por esse motivo, torna-se necessário o uso de espaços de memória para armazenar instruções e dados temporários. O sistema de I/O é o responsável por realizar a comunicação entre o sistema e o mundo externo. A troca de informação entre as portas de I/O, processador e a memória é realizada por meio dos barramentos.

Esses componentes geralmente estão organizados em uma arquitetura. A arquitetura mais comum é a de von Neumann. Neste tipo de arquitetura, os programas/instruções são armazenados no mesmo espaço da memória de dados. O endereçamento da memória é feito por posição (endereço) e não pelo tipo, além de que as instruções são executadas sequencialmente. A máquina proposta por John von Neumann é composta pelos quatro componentes citados. A Figura 2a mostra o esquema básico da arquitetura proposta.



(a) - Esquema da arquitetura de von Neumann

(b) - Esquema da arquitetura de Harvard

Figura 2: Esquemas das arquiteturas de von Neumann e Harvard

Vindo da necessidade de se trabalhar de forma mais rápida do que a arquitetura von Neumann (AV) possibilita, uma adaptação foi proposta chamada de arquitetura Harvard (AH). Ela se difere da de von Neumann no ponto em que há duas memórias separadas:

uma para instruções e outra para dados. A vantagem por trás dessa separação é a capacidade dada ao processador de buscar uma nova instrução enquanto executa outra. Isso é possível pois enquanto está ocorrendo o processamento de uma instrução, somente o barramento processador/memória de dados está sendo usado, deixando livre o barramento processador/memória de instruções está livre (Figura 2b). Já na AV, há somente um barramento para os dados e instruções percorrerem, o qual faz com que somente uma instrução seja processada por vez, tornando o processamento mais lento quando comparado com a de Harvard.

A AV, quando comparada a de AH, é mais simples por apresentar somente uma memória. Ela não permite acesso simultâneo às memórias, além de não permitir *Pipeline* (técnica de hardware que permite o processador realizar a busca de uma ou mais instruções, além da próxima instrução a ser executada - Figura 3), o que a torna mais lenta. O pipeline é usado para acelerar a velocidade de operação do processador, já que a próxima instrução a ser executada já foi buscada e está no processador. Isso faz com que não haja “perda de tempo” ao buscar na memória a próxima instrução e evita que o processador fique ocioso durante esse tempo. A sua memória é particionada em espaços do tamanho de um endereço (memória linear), e por esse motivo é necessário um mapa de memória. Em sistemas x86/x86-64, o mapeamento de I/O é feito de forma diferente chamado de PORTED I/O. Essa separação facilita o processo de endereçamento, pois toda a complexidade de I/O é descartado do projeto processador/memória, fazendo com a processador tenha uma lógica interna menor, mais barata, rápida e simples de ser construída. A AH desvincula o tamanho da instrução do tamanho da palavra de dados/endereço.

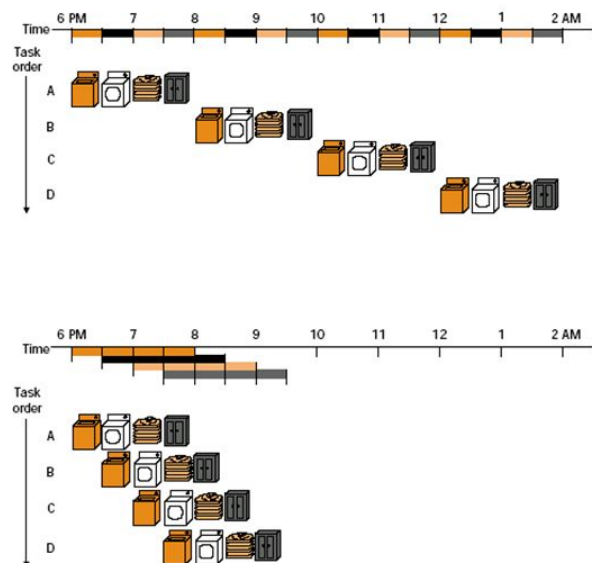


Figura 3: Ideia por trás do pipeline

A arquitetura von Neumann é presente na maioria dos microprocessadores tradicionais encontrados no mercado, enquanto a arquitetura de Harvard é usada pela grande maioria dos microcontroladores e DSPs, dado o uso deles em sistemas de tempo real.

A construção de qualquer arquitetura para um MC/MP/DSP envolve o projeto do conjunto de instruções que será usado, tendo RISC e CISC como as principais abordagens. Os

computadores RISC partem do pressuposto que um conjunto mais simples de instruções resultará em um processador mais simples, barato e rápido. No caso das arquiteturas CISC, o foco é facilitar a construção dos compiladores criando arquiteturas complexas o bastante para tal. Dessa forma, programas complexos são compilados em programas perto da linguagem de máquina mais curtos. Com programas mais curtos, os sistemas digitais CISC necessitam acessar um número menor de vezes a memória para buscar instruções. Isso para torná-los mais rápidos. As duas arquiteturas podem ser definidas como:

- **CISC** (Complex Instruction Set Computing) - define um conjunto complexo de operações usando somente algumas instruções. Como consequência, apresenta um código menor, mas com um hardware mais sofisticado. O resultado é um chip mais caro. Geralmente, arquiteturas baseadas em von Neumann utilizam CISC. Exemplo: Intel 4004 (46 instruções), 8080 (78 instruções), 8085 (150 instruções) e Z80 (mais de 500 instruções).
- **RISC** (Reduced Instruction Set Computing) - define um conjunto mais simples de instruções, o que resulta em uma arquitetura mais simples com menos operações feitas com menos instruções. Em comparação com a CISC (operações de múltiplos ciclos), a RISC apresenta um número reduzido de ciclos de clock por operação disponível (operações com um ciclo). Por esse motivo, apresenta um Hardware mais simples (chip mais barato), contudo a complexidade para compilar o código é maior e são necessários uma otimização dos pipelines de execução. Geralmente, arquiteturas baseadas em Harvard utilizam RISC. Exemplo: Intel 8086, 8088 e o PIC (35 instruções).

Com o passar dos anos, os processadores RISC têm se tornado mais complexos e reduzido o número de instruções, enquanto os processadores CISC, mais eficientes e passaram a utilizar pipeline. Por esse motivo, está ficando cada vez mais complexo a diferenciação de RISC e CISC. Além disso há alguns processadores que usam o conceito de ambos, tendo em seu *core* um RISC e ter uma arquitetura CISC. A Tabela 1 sumariza as principais características das duas arquiteturas.

Características	RISC	CISC
Arquitetura de Memória	Registrador-Registrador	Registrador-Memória
Tipos de Dados	Pouca variedade	Muito variada
Formato das Instruções	Inst. poucos endereços	Inst. com muitos endereços
Modo de Endereçamento	Pouca variedade	Muita variedade
Estágios de Pipeline	Entre 4 e 10	Entre 20 e 30
Acesso aos dados	Via registradores	Via memória

Tabela 1: Comparação arquitetura CISC e RISC

Baseado na tabela 1, é possível notar as diferenças entre as arquiteturas CISC e RISC. A RISC visa um processador o mais simples possível e rápido, e acaba-se por utilizar um

número menor de instruções, com pouca variedade e com poucos endereços. Uma vez que o número de instruções é menor, fica mais fácil para o processador designar o número de ciclos necessários para executá-lo, o que favorece muito o Pipeline. Nesse caso, o processador conseguirá saber quando será possível iniciar o estágio da próxima instrução. Visando a velocidade e a quantidade de instruções/dados, a arquitetura registrador-registrador é utilizada. Ao contrário da RISC, a arquitetura CISC foca em processadores poderosos e capazes de executar tarefas mais complexas. Atualmente, as arquiteturas CISC oferecem a oportunidade de analisar uma sequência de instruções ao mesmo tempo, por meio da execução fora de ordem (processador analisa uma sequência de instruções simultaneamente, processando as instruções que não possuem dependência entre si) e a execução superescalar (organiza o processador em diversas unidades de execução específicas, as quais trabalham simultaneamente - unidade de inteiros, unidade de pontos flutuantes). Há a possibilidade da execução de ambos, onde instruções que não estejam na sequência de execução podem ser executadas para evitar que as unidades de execução fiquem ociosas. O CISC acaba por usar a estrutura registrador-memória já que trabalha com um conjunto maior de instruções mais complexos, os quais não podem ser armazenados em registradores devido à sua grande quantidade.

Um exemplo de microcontrolador com AH é o PIC (Figura 2), o qual utiliza arquitetura RISC. Os PICs são microcontroladores fabricados pela Microchip Technology, capazes de processar dados de 8 bits, 16 bits e, recentemente, 32 bits. Já o tamanho das instruções usadas pelo núcleo de processamento é de 12 bits, 14 bits e 16 bits. Eles trabalham com frequências de 4 MHz até 48 MHz, usando ciclos de instruções mínimo de 4 períodos de clock, o que resulta em uma velocidade máxima de 10 MIPS (milhões de instruções por segundo). O PIC faz reconhecimento de interrupções tanto externas quanto internas. A tensão de alimentação é de 2 a 6V, e há diversos modelos de encapsulamento de 6 a 100 pinos em diversos formatos (SOT23, DIP, SOIC, TQFP, PLCC). O SFR (Special Function Register) é um registrador do processador, o qual controla e monitora vários aspectos das funções do processador. Entre suas atribuições, destacam-se: controle de periféricos de I/O, temporizador, ponteiro de pilha, limita de pilha (evitar estouro), controlador de programa, endereço de retorno de subrotina, status do processador (manutenção de uma interrupção, execução em modo protegido) entre outros. Todas estas atribuições são dependentes da arquitetura do processador.

Alguns sistemas de tempo real utilizam DSPs para o processamento de sinais. Nesse tipo de cenário, o tempo de resposta pré-determinado deve ser satisfeito, caso contrário pode causar grandes danos, como o não acionamento de um airbag no momento da colisão de um carro, o que aumenta as chances de morte do usuário. Considerando esse contexto, DSPs tendem a usar a arquitetura Harvard para que o processamento seja o mais rápido possível, visando o processamento de tempo real.

Ambas as arquiteturas compartilham os 4 componentes citados e, juntos, compõem o circuito interno de um microprocessador/microcontrolador e DSP. Cada um com sua função, desde buscar um programa e executá-lo, buscar e salvar dados, até comunicar com componentes fora do chip.

A programação de MPs e DSPs podem tanto ocorrer por meio de Assembly (linguagem próxima a das instruções dos processadores com a correspondência de uma para um entre instruções simbólicas e códigos de máquina executáveis), como por meio de outras linguagens que incluem recursos de baixo nível, como C. No caso de linguagens de alto nível, o código será compilado e substituído por instruções, as variáveis por códigos binários e endereços de memória correspondentes. Algumas linguagens vêm sendo atualizadas para incluir mecanismos que permitam o seu uso em sistema de tempo real, como o Java. O mecanismo mais básico usado são as threads (instância de uma tarefa em um sistema operacional).

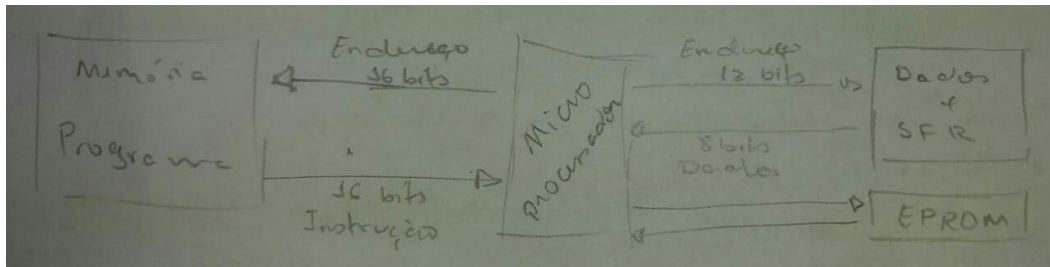


Figura 2 - Arquitetura típica de um PIC

Cada família de MPs e DSPs tem sua própria linguagem de programação e suas próprias ferramentas de desenvolvimentos, os quais são fornecidos pelos fabricantes. Entretanto, há ferramentas como Matlab/Simulink e LabView, que em conjunto com os compiladores adequados, oferecem a opção de criação de projetos em DSP usando diagramas de blocos.

Um microcontrolador bastante usado tanto na indústria, quanto na academia, é o PIC16F27A. Ele possui um conjunto reduzido de 37 instruções e a memória de programa com largura de 14 bits. O endereço de cada registrador pode ser de até 8 bits, o que faz com que não seja possível usar dois registradores na mesma instrução.

Pode-se dividir as instruções do PIC16F27A em algumas categorias: instruções de movimentação de dados, cálculos aritméticos (soma, subtração), operações lógicas (e, ou, ou exclusivo), operações de atribuição (incremento, decremento, rotação de bits), de retorno de subrotina, funções de pular instruções com incremento e desvios condicionais. Estas podem ainda serem agrupadas em cinco grupos genéricos:

1. Operações no registrador de trabalho (WREG) e operando de 8 bits. Por exemplo, movimentação de literal, operações lógicas e até a instrução de retorno, a qual carrega imediatamente o literal em WREG e retorna (desvios computadorizados)
2. Operação com o WREG ou outro registrador selecionado. Nesse caso, o resultado de operação aritmética pode ser salvo em um registrador, seja o WREG ou algum outro específico.
3. Operações em bits. Dado um número do registrador e um número de bit, pode-se realizar quatro operações: definir ou limpar um bit, e testar e pular na definição/limpeza (usada em desvios condicionais).
4. Mecanismos de controle de fluxo do programa. Instruções de salto dentro do código.

5. Instruções sem um operando específico. Por exemplo, retorno de subrotina e sleep para entrar em modo de economia de energia.

Os mecanismos de controle de fluxo são dois: (i) *goto* que muda o fluxo baseado em um endereço determinado; e (ii) *call* que muda o fluxo baseado em um endereço determinado e salva em um registrador de endereço de onde foi a origem da chamada (salva na pilha de chamada). A pilha de chamada dos PICs são implementadas em hardware, a qual é utilizada para guardar endereços de retorno após o uso de calls. Muitas funções em alto nível são implementadas desta forma. A pilha é limitada em 8 bytes. Se caso ultrapassar os 8 bytes ocorre o *stack overflow*.

Baseado em todas as instruções citadas, os códigos são criados. Apesar de ser possível criar o mesmo código de alto nível com o Assembly, o código criado com Assembly é muito verboso, onde cada operação deve ser descrita passo a passo, além de serem mais difíceis de serem usados, debugados e interpretados. Considerando esse cenário, linguagens de mais alto nível, como por exemplo o C, que incluem recursos de baixo nível começaram a ser utilizadas. O código Assembly tende a ser mais rápidos do que os códigos criados em linguagens alto nível.

Um das vantagens do C em comparação a outras linguagens de alto nível está relacionada ao ponto que os compiladores de C (programas usados para traduzir o código alto nível para instruções mais próximos da linguagem de máquina) são geralmente muito eficientes. Há fabricantes que afirmam que o código em linguagem de máquina compilado de um programa em C é no máximo 20% maior do que o se fosse escrito em Assembly. Na prática a eficiência dos compiladores ficam na faixa de 50% a 70%. Quanto maior o código gerado, maior o número de instruções e mais lento a execução.

O programa criado em C para um PIC se assemelha muito aos programas criados para outras aplicações. A diferença é o uso de bibliotecas específicas, como a biblioteca *pic16f62xa.h* (PIC16F27A), a qual introduz os conceitos do PIC para a linguagem C. Por exemplo, há a inclusão de TRISx (configura o sentido do fluxo de dados da porta x, onde no caso do PIC16F27A, são as portas A e B - fluxo zero saída e um entrada), PORTx (escreve e lê o nível dos pinos na porta x), e PORTxbits. PORTxy (x é qual a porta e y qual o bit específico a serem acessados), além de outras.

As aplicações tanto para MCs e DSP são diversas, desde automação e controle de processos em geral, até TV digital, e smartphones. O seu uso envolve um planejamento tanto do hardware quanto do software. Por exemplo, a forma de se trabalhar com um DSP, o qual vem vazio de fábrica (software), envolve as seguintes etapas:

1. Projetar o programa que será executado pelo DSP
  - Para alguns a linguagem usada é exclusivamente o Assembler (família CZXX e C5XX de DSPs da Texas), outros já permitem o C (PIC).
  - Aplicação de algoritmos aos sinais convertidos do PIC, os quais são representados em forma de funções matemáticas (Transformada de Fourier), para que se possa trabalhar com eles internamente.
  - Alguns fabricantes já fornecem o código da aplicação desejada pronto aos desenvolvedores.

2. Utilização de uma interface para salvar o programa desenvolvido para o DSP
  - Conecta o computador ao DSP e serve para passar a memória de programa.
3. Montagem do protótipo
  - Envolve testar e adaptar o DSP as necessidades do ambiente que o cerca. Por exemplo, ao trabalhar com sinal de áudio é necessário garantir que a intensidade necessária chegue ao conversor D/A. O mesmo para um sinal de saída que deve ser adequado ao ambiente que o envolve.
  - Verificação da necessidade de memória extra para o DSP e o que fazer se caso for insuficiente: circuito de aviso ou memória externa.

Tanto o MC, quanto o DSP (MP tem um uso com foco diferente), vêm possibilitando novas áreas de se expandirem, como por exemplo os sistemas ciber-físico (Cyber Physical Systems - CPF). Os CPFs são a convergência entre computação, comunicação e controle, sendo a integração da computação com processos físicos. Acaba-se por utilizar MC/DSP para interagir com processos físicos, os quais adicionam novas propriedades ao sistema.

Os MCs e DSPs realizam todo o processamento e interface entre o mundo e o sistema digital de um CPF. São eles que oferecem a base necessária para o CPF desenvolver e também a Internet-das-coisas. Isso ocorre pois eles estão presentes em praticamente todos os dispositivos eletrônicos digitais que são usados diariamente, além de permitir a evolução de equipamentos que há anos não evoluíam (motores a combustão que atualmente podem funcionar com sistema bicomustível com um menor nível de poluição devido ao controle eletrônico empregado). O CPF está um nível abaixo da Internet-das-coisas.

No contexto de Internet-das-coisas, os CPFs possuem um papel de destaque, uma vez que colaboram para construir sistemas inteligentes, como por exemplo as cidades inteligentes, as quais podem ser vistas como sistemas ciber-físicos em ampla escala. As cidades inteligentes possuem sensores monitorando indicadores virtuais e físicos e por meio de atuadores, muda dinamicamente o ambiente urbano complexo de alguma maneira. Os CPFs podem ser usados na produção a âmbito industrial para construir arquiteturas baseadas na internet que facilitam o controle remoto de sistemas de produção *stand-alone*. Os CPFs vêm se destacando em vários campos, desde governos, organizações até indústrias de tecnologia, os quais direcionam seus esforços para integrar diversos sistemas em redes inteligentes, a fim de, por exemplo, controlar uma rede de energia ou o tráfego de trânsito para torná-lo mais seguro.

Um ponto de convergência entre MC e DSP é o seu uso especializado e que atualmente está presente no dia-a-dia do ser humano. Para o desenvolvimento de sistemas com MC/DSP, torna-se necessário o entendimento de sua arquitetura, que em sua maioria são Harvard devida aos seus ganhos quando comparado a de von Neumann, e os componentes que os compõem, além da programação envolvida no processo. Baseado nesses pontos, pode-se criar sistemas mais complexos, eficientes e adequados para uma situação específica.

Disponível em: [https://github.com/plsilva/Files\\_DigitalSystems](https://github.com/plsilva/Files_DigitalSystems)

<http://producao.virtual.ufpb.br/books/edusantana/introducao-a-arquitetura-de-computadores-livro/livro/livro.chunked/ch04s04.html>