Arquitetura de Unidade Funcionais Computacionais

Microprocessadores, sistemas de memória, sistemas de I/O e barramentos

Dentre os diversos elementos presentes no hardware de sistemas computacionais, incluindo Sistemas Embarcados, destacam-se quatro: processadores, sistema de memória, sistema de entrada e saída (I/O) e barramentos. É extrema a importância de entender-se esses quatro componentes para que se compreenda a arquitetura de um sistema, uma vez que o potencial e a capacidade do mesmo é dependente desses quatro, além de outros componentes.

A arquitetura de um computador em geral (incluindo Sistemas Embarcados - SE) deve ser capaz de executar uma série de instruções em sequência. Essas instruções serão processados pelo processador/CPU (Central Process Unit) uma a uma, em uma ordem pré-estabelecida. Elas serão executadas em ciclos de clock, sendo que a cada pulso, parte da execução da instrução será feita. A CPU só 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. A origem das instruções e dos dados podem tanto ser internos, quanto de I/O. O sistema de I/O realiza a interface entre o restante do chip e a CPU, mais a memória usada por ela. Toda a troca de informação, mesmo dentro do próprio Chip (CPU <-> Memória) é feita por meio de barramentos. As etapas de execução de uma instrução dentro da CPU pode ser vista da seguinte forma:

IR -> ID -> AG -> DF -> GX -> WB

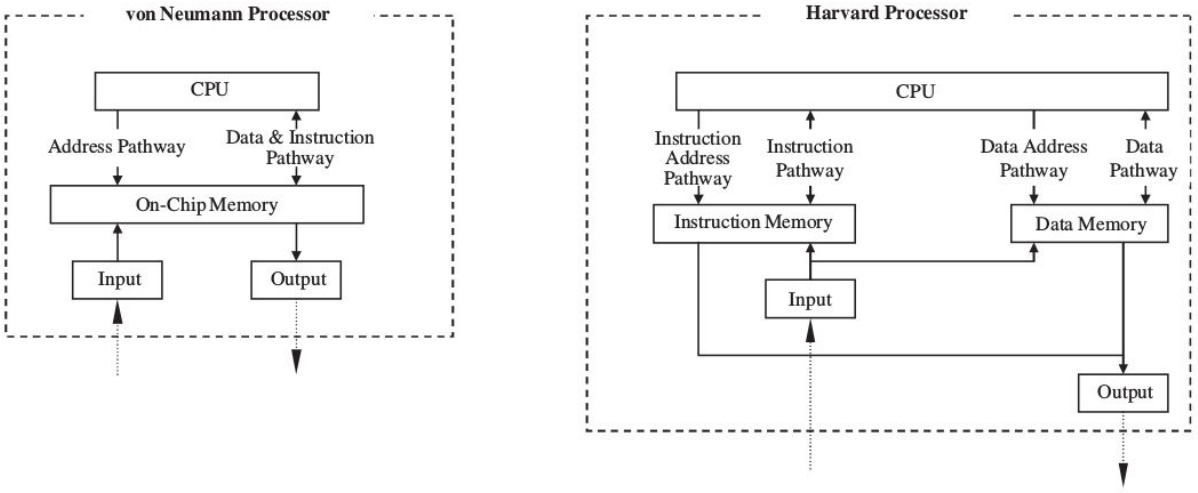
onde:

- IR -> Busca a próxima instrução
- ID -> Decodifica instrução
- AG -> Descobre o endereço na memória de dados (registradores)
- DF -> Transfere os dados para registradores
- EX -> Executa a ação na ULA (Unidade Lógica Aritmética)
- WB -> Escreve de volta nos registradores o resultado

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. A máquina proposta por John von Neumann é composta pelos quatro componentes citados. A Figura 1a mostra o esquema básico da arquitetura proposta.

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 CPU/memória de dados está sendo usada, deixando livre o barramento CPU/memória de instruções está livre (Figura 1b). 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, isso considerando ambas com o mesmo conjunto de instruções e a mesma frequência de relógio (clock).



(a) - Esquema da arquitetura de von Neumann

(b) - Esquema da arquitetura de Harvard

Figura 1: Esquemas das arquiteturas de von Neumann e 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 *Pipelining* (parte de instruções serem executadas de cada vez), o que a torna menos eficiente. A sua memória é particionada em espaços do tamanho de um endereço (memória linear), 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 pois toda a complexidade de I/O é descartado do projeto CPU/memória, fazendo com a CPU 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.

Geralmente, AV utiliza instruções CISC (hardware mais complexo e software mais simples), enquanto AH, RISC (hardware mais simples e software mais complexo). Ambos serão abordados nos parágrafos a seguir.

Para ilustrar um microcontrolador com AH, abordarei arquitetura dos PICs (Figura 2) por serem amplamente populares. Entretanto, salienta-se que há outros microcontroladores de outros fabricantes, como os microcontroladores de outros fabricantes, como os microcontroladores da Atmel e também Texas Instruments, os quais são AH e

predominantemente RISC. Os PIC's 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 a partir de 4 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). No projeto do PIC, destaco o SFT (Special Function Register) é um registrador do processador, o qual controla e monitora vários aspectos das funções do processador. Entre as atribuições do SFT, 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.

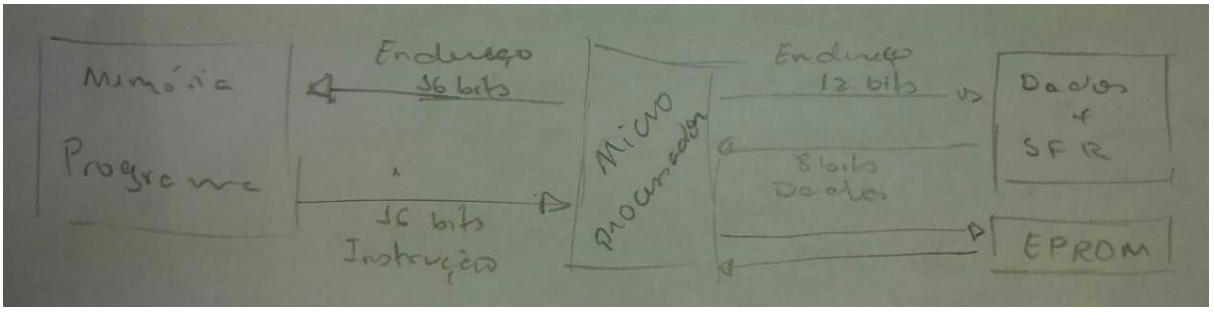


Figura 2 - Arquitetura típica de um PIC

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

Geralmente, os componentes são construídos a partir de componentes de sistemas digitais, como portas lógicas, decodificadores, multiplexadores e até Latches e Flip-Flops (memória RAM). A implementação começou com relés eletromagnéticos, passando pelas válvulas, até os semicondutores (transistores e microprocessadores - MP).

O MP contém a CPU e um conjunto limitado de memória e I/O. Normalmente, eles não possuem memória RAM (Random Access Memory) ou ROM (Read-Only Memory) dentro da pastilha (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 desenvolvimento de software, websites, editor de fotos. Nesse caso não há uma relação forte entre a entrada e saída.

O processador de um MP é a unidade principal de uma placa de um sistema embarcado e sua principal função é a de processar instruções e dados. Um SE possui pelo menos um processador principal responsável pelo controle central do dispositivo, e outros

processadores “escravos”, os quais são controlados pelo principal. Esse cenário é bastante comum em Processadores Digitais de Sinais (DSP - Digital Signal Processors). Os processadores escravos podem tanto estender as instruções do principal, quanto servir como controlador de memória, barramentos ou I/O. No caso do Ampro’s Encore 400 Board, por exemplo, há um processador principal (dentro da *Host-Peripheral Interface*) e controladores (processadores escravos) de ethernet e de I/O (Figura 3).

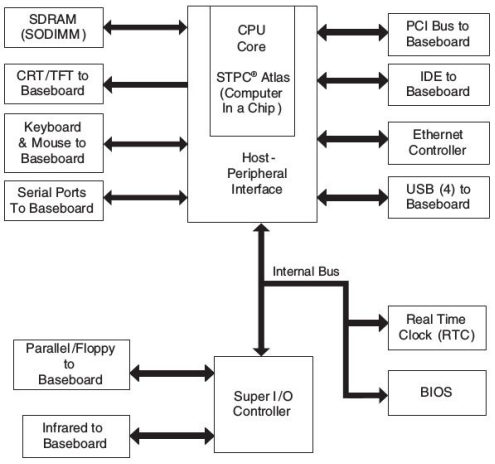


Figura 3: Ampro’s Encore 400 board

O mercado oferece diversas arquiteturas de SE baseados na arquitetura de von Neumann, incluindo a de Harvard. Cada uma delas possui um conjunto de instruções e um conjunto de características baseados na arquitetura. Essas características são geralmente conhecida como *Instruction Set Architecture*, ou ISA. É necessário definir os **operandos** (dados), as **operações** definidas para manipulá-los, a **forma de armazenamento**, os **modos de endereçamento** usados para ganhar acesso e processar os operandos, e a **forma de lidar com interrupções**. A implementação da ISA é um fator determinante para definir as características de design de um sistema digital, uma vez que impactam na performance, tempo de design e planejamento, funções disponíveis e o custo.

Todas as **operações** definidas na ISA são constituídas por uma ou mais instruções. Para cada processador, pode ser definida um número, tipos de instruções e nome de operações diferentes com o mesmo propósito (mesma operação). Isso ocorre pois apesar de ter as mesmas operações, os processadores possuem diferentes grupos de instruções. Ficam definidos na ISA os tipos e formatos das operações. As operações geralmente incluem computações na ULA, movimentação de dados, ramificação (movimentação condicionais/ incondicionais para outra área do código para processar - *goto*), operações de I/O, e operações de troca de contexto (onde a informação do registrador de localização é temporariamente salvo, quando há alguma troca para a execução de uma rotina e ao seu fim, o contexto é trocado novamente e a instrução original é executada).

O **formato das operações** é na verdade o número de combinações dos bits (1’s e 0’s) que representa uma operação, conhecida como código de operação ou opcode. O MPC823 possui instruções de 6 bits long (0-63 decimal), enquanto MIPS32/MIPS o opcode é de 6 bits long, contudo de opcode pode variar de acordo com a sua localização.

Os tipos e formatos dos **operandos** (dados) são definidos também na ISA. A forma mais comum e simples das arquiteturas definirem os tipos são *bytes* (8 bits), *half-words* (16 bits) e *words* (32 bits). Os tipos mais complexos como inteiros, caracteres e pontos flutuantes são baseados nos tipos mais simples. Os tipos binário, decimal e hexadecimal que a arquitetura irá suportar também são definidos na ISA. A forma como os operandos são salvos também é definida na ISA. Cada processador (que possui sua própria ISA) possui diferentes definições de onde os dados são salvos, além de como os dados são salvos (incluindo a ordem como os bits de um byte são salvos). As duas formas de ordenação de bytes: **big-endian** (bit mais significativo salvo primeiro - 68000 e SPARC) e **little-endian** (bit menos significativo é salvo primeiro - x86).

As **formas de endereçamento** define a forma como o processador vai acessar a área de armazenamento dos operandos. As duas formas mais comuns de endereçamento são: (i) **arquitetura load-store**: só permite operações com dados dos registradores e em mais nenhum outro tipo ou lugar da memória; e (ii) **arquitetura register-memory**: permite operações com dados dos registradores e outros tipos de memória.

Outro ponto importante é a forma como o processador **lida com interrupções e exceções**. Ambos são mecanismos usados para parar o fluxo normal de um programa, e começar a executar um novo conjunto de operações. Isso só ocorre para responder a um evento, como problemas de hardware, resets, entre outros.

Os modelos de ISA mais comuns de serem implementadas são a de aplicações específicas (define o processador para aplicações específicas de embarcados, como modelos de controle e a JVM), de uso geral, paralelismo a nível de instrução (múltiplas instruções executadas em paralelo - uma evolução do RISC), ou alguma implementação híbrida. Dentre estas, destaca-se a implementação de uso geral a qual apresenta um uso em uma grande gama de sistemas, ao invés de SE específicos. As arquiteturas mais simples implementadas são:

- **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. Exemplo: Intel 8086, 8088 e o PIC (35 instruções).

Com o passar dos anos, os processadores RISC têm se tornado mais complexos, enquanto os processadores CISC, mais eficientes. 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 **CPU** de um processador é a unidade de processamento, responsável por executar o ciclo de buscar, decodificar, e executar instruções. Esse ciclo é implementado usando quatro componentes básicos: ULA, registradores, Unidade de Controle (UC) e os barramentos internos da CPU. A Unidade de Controle possui a função semelhante a Máquina de Turing Universal: buscar um programa em memória, instrução por instrução, e executá-las sobre um conjunto de dados de entrada. Os registradores são usados para armazenar os dados mais recentes ou os mais usados (cache). Já a ULA é a responsável por executar as operações definidas na ISA. Os barramentos internos da CPU são os responsáveis por conectar a ULA, registradores e a UC.

Atualmente, a maioria dos processadores oferecem alguma forma de paralelismo, o qual afeta o tempo de execução de um programa. A concorrência é comum em sistemas paralelos, e ocorre quando diferentes programas (ou do mesmo programa) executam simultaneamente. O intuito de usar o paralelismo, em geral, é de acelerar o processamento, contudo, no contexto de SE, a concorrência é mais vital, uma vez que os programas para embarcados interagem com processos físicos externos a eles, os quais podem ocorrer todos ao mesmo tempo. Os SEs normalmente monitoram e reagem a concorrência de múltiplas fontes externas, e simultaneamente, controla múltiplos processamentos e dispositivos de saída. O paralelismo pode ocorrer de duas formas: arquiteturas multicore (vários processadores dentro de um mesmo chip), ou paralelismo a nível de instrução (o processador é capaz de executar múltiplas operações independentes em cada ciclo de instrução).

Esse cenário de paralelismo se torna importante no contexto de **interrupções**. Interrupções são sinais disparados por algum evento durante a execução de uma instrução do processador principal. Elas podem ocorrer tanto de forma síncrona (ocorre devido as instruções), como assíncrona (quando ocorre externo ao processador). O custo de parar o processamento completo do processador principal para analisar e resolver a interrupção, pode ser danoso para o restante do processamento do SE. As interrupções podem ser de três tipos: software, hardware interno, e hardware externo. Para interrupções de hardware externo, é usado um pin interno ligado ao processador mestre, chamado de IRQ (**Interrupt Request Level**).

Os sistemas de computação, incluindo os SE, possuem uma hierarquia de memória (Figura 4). É uma coleção de diferentes tipos de memória, cada uma com velocidade, tamanhos e usos diferentes. Algumas memórias podem ser fisicamente integradas ao processador, como registradores, e alguns tipos de memória primária, a qual é uma memória conectada/integrada diretamente no processador, como a memória ROM (Read Only Memory), a RAM (Random Access Memory) e a cache level-1.

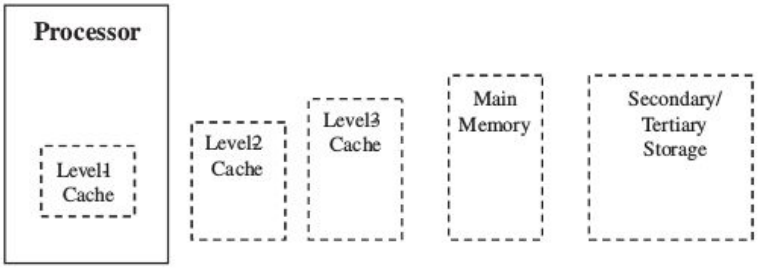


Figura 4: Hierarquia de memória

A **cache level-1** é uma memória que está dentro da CPU. É usada como cache. O custo das memórias usadas na cache são mais caras devido a sua velocidade, é por isso que normalmente são pequenas. A cache level-1 é constituída por registradores, ou seja, flip-flops/latchs. Ela armazena um subconjunto de dados da memória principal usados recente e frequentemente. Como são armazenadas cópias, há a ilusão de que o processador principal que está operando com dados da memória principal, contudo, ele na verdade está executando da cache. Devido a capacidade baixa da cache e o tempo gasto de buscar uma instrução na memória, foi necessário a criação de políticas de substituição dos dados da cache. As mais comuns são: *Random Replacement*, *First-In-First-Out*, *Last-In-First-Out* e *Last Recent Used*.

A **memória cache level-2+** (ou mais, incluindo a level-3 da Figura 5), possui todas as características da cache level-1, contudo ela fica entre a CPU e a memória principal (RAM) e constituída com memória SRAM (mais lenta que registrador e mais barata).

A memória primária, geralmente, é parte da memória do subsistema e é constituída por três componentes: a memória do circuito integrado, barramento de endereço e o barramento de dados. É mostrado na Figura 5, a junção destes componentes.

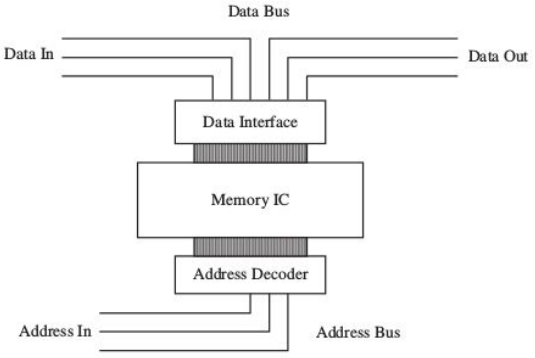


Figura 5: Hardware da memória primária do subsistema

A memória do circuito integrado (IC) é composta de três unidades: array de memória, decodificador de endereço e interface de dados. O array de memória é a memória física que armazena os dados. Apesar do processador principal e do programador tratarem a memória como um array unidimensional (cada célula do array é uma linha de bytes e o número de bits por linha pode variar), na realidade a memória física é uma matriz bidimensional (array com duas dimensões) constituída de células de memória endereçadas por uma única linha

e coluna. Cada célula pode guardar somente 1 bit. O endereço de cada célula usando a memória como uma matriz bidimensional, por meio de linha e coluna, é conhecido como endereços de memória física. O *address decoder* (decodificador de endereço) pega o endereço pedido, converte em um endereço físico e busca na memória. A interface de dados, baseado no endereço, os dados para o barramento de dados para a transmissão.

As memórias primárias e secundárias podem ser divididas em dois grupos: voláteis e não-voláteis. Na primeira, a memória perde todos os dados (bits) quando a fonte de alimentação da placa é desligada. Enquanto na segunda, mesmo quando a fonte de alimentação é desligada, os dados ainda continuam armazenados. Nos SE, há dois tipos de famílias de memórias não-voláteis (ROM e memória auxiliar), e há um de memória volátil (RAM).

A memória **ROM** é usada para armazenar dados em SE permanentemente, uma vez que ela é não-volátil. Normalmente, os tipos de dados salvos nela são os softwares que o dispositivo necessita para funcionar depois de ter sido enviado para fora da fábrica. O conteúdo da ROM pode ser lido somente pelo processador principal, contudo, dependendo da ROM, o processador principal será capaz de editá-la ou até mesmo apagá-la. Os tipos mais comuns de ROM são: *ROM, PROM, EPROM e EEPROM/Flash.* Os programas de SE são armazenados de duas formas: ou na **EPROM**, ou **FLASH**.

Um bom exemplo de memória ROM é a BIOS. O BIOS (um acrônimo Basic Input/Output System e também conhecido como System BIOS, ROM BIOS ou PC BIOS) é um tipo de firmware usado para realizar a inicialização do hardware durante o processo de inicialização em computadores, e para fornecer serviços de tempo de execução para sistemas operacionais e programas. O BIOS é um programa de computador pré-gravado em memória ROM (firmware) executado por um computador quando ligado. Ele é responsável pelo suporte básico de acesso ao hardware, bem como por iniciar a carga do sistema operacional.

A memória **RAM**, conhecida como Memória Principal, é utilizada em SE para armazenar dados temporários que podem ser alterados mais de uma vez, contudo eles são perdidos quando a energia é cortada (memória volátil). Há dois tipos de RAM: *static RAM* (SRAM) e a *dynamic RAM* (DRAM). A tecnologia da SRAM é baseada em transistores, o que faz com que ela mantenha os dados sem a necessidade de serem reescritos. Já a tecnologia da DRAM é baseada em capacitores, o que faz com ela perca os dados conforme o tempo vai passando (fator capacitivo), necessitando, assim que os dados sejam reescritos. Esse fator torna a DRAM mais lenta que a SRAM, além de consumir mais energia, e acaba por ser mais barata.

O **pipeline ao buscar uma informação** pode ser descrito da seguinte forma: Antes de buscar qualquer informação na memória principal que o processador precisa, primeiro busca-se na cache level-1, se estiver (*cache hit*), o dado é retornado para a CPU, se não estiver (*cache miss*), busca-se na próxima cache. Se o dado estiver na cache level-2 (*cache hit*) salva-se na cache level-1 e retorna para a CPU, se não estiver (*cache miss*), busca-se na próxima cache. Se ao fim não estiver em nenhuma cache, busca-se na memória principal e vem salvando nas caches. Esse processo de escrita pode ocorrer de duas formas: *write-through* (os dados são escritos em todas as caches e na memória principal

toda vez) ou *write-back* (os dados são escritos somente na cache, e somente será escrito na memória principal se o dado sair da cache).

As **memórias auxiliares**, também chamadas de memória secundária, são aquelas que estão conectadas ao processador principal por meio de outro dispositivo. É uma memória usada em sua maioria para armazenamento de dados (são grandes), são memórias não-voláteis e são classificadas como dispositivos de I/O. O seu acesso só pode ser feito por um dispositivo ligado ao SE, por isso é uma memória mais lenta que a principal (RAM) e é usada somente em último caso. Os exemplos mais comuns de memória secundária são: fitas magnéticas e disco rígido (Hard Drive - HD).

Como é possível haver diferentes tipos de memória dentro de um mesmo sistema, faz-se necessário um gerenciador de memória (*memory managers*). Em alguns casos ele é integrado ao processador principal. Os dois tipos mais comuns de controladores são: controlador de memória (*memory controllers - MEMC*) e as unidades de gerenciamento de memória (*memory management units - MMUs*). O MEMC é o responsável por implementar e fornecer interfaces para diferentes tipos de memória no sistema (SRAM e DRAM), sincronizando o acesso a memória e verificando a integridade dos dados transferidos. Já a MMU permite a flexibilidade do sistema em ter uma grande memória virtual com uma pequena memória física. Ela pode existir fora do processador principal e é usada para traduzir um endereço lógico (virtual) para um endereço físico (mapa da memória), além de lidar com a segurança/proteção da memória, controlar a cache, gerenciar o barramento entre a CPU e a memória, e gerar exceções apropriadas (interrupções).

Em sistemas onde o MMU realiza o parse do endereço de memória, a cache pode ser integrada entre o processador principal e o MMU, ou MMU e a memória principal. Em ambos os casos há vantagens e desvantagens, a maioria relacionada a **DMA** (*direct memory access*). O DMA é o acesso direto de um processadores escravos a memória principal sem passar pelo processador principal. Quando a cache está integrada entre o processador principal e MMU, pode ocorrer uma inconsistência entre a memória principal e a cache, uma vez que somente o processador principal altera a cache e a memória principal será alterada por outro processador escravo sem ser o processador principal, deixando os dados da cache e da memória principal diferentes. Uma solução é o acesso do processador principal a memória somente quando os dados da DMA estiverem sendo transferidos ou se a cache for mantida atualizada por unidades do sistema além do processador principal. Quando a cache é integrada entre a MMU e a memória principal, torna-se necessário mais traduções de endereços, uma vez que a cache é utilizada tanto pelo processador principal quanto pela DMA.

Quando sistema precisa comunicar com o ambiente, ele faz-uso do sistema de I/O. Os componentes que compõem o sistema de I/O são responsáveis por transmitir a informação para dentro e para fora do sistema embarcado e até mesmo no contexto do processador. O sistema de I/O tem componentes de entrada (traz a informação de um dispositivo de input para o processador principal) e os componentes de saída (leva a informação do processador principal para algum dispositivo de saída) e componentes que realiza ambas as funções. A conexão desses dispositivos pode tanto ser por cabos ou wireless (desde que

tenha uma interface com o SE, seja por bluetooth ou usb gadget usados por alguns mouses e teclados sem fio).

Os componentes de I/O podem ser divididos em algumas categorias (o mesmo componente pode estar em mais de uma categoria):

- Comunicação I/O e rede (a camada física do modelo de redes usado atualmente, o OSI - Open Systems Interconnection).
- Entrada (teclado, mouse, controle remoto).
- Saída de I/O e telas (touch screen, impressoras, LEDs).
- Armazenamento (controlador de disco óptico, controlador de disco magnético).
- Debugging I/O (porta serial e porta paralela).
- Tempo-real e I/O diversos (temporizadores/contadores, conversores AD e DA, chaves seletoras).

A forma de transmissão entre o processador principal e o dispositivo I/O pode ocorrer de duas formas: serial e paralela. A primeira consiste no armazenamento, transferência e recebimento de um bit por vez, além disso os seus subsistemas necessitam de uma porta serial e uma interface serial. Já a segunda, consiste no armazenamento, transferência e recebimento de vários bits simultaneamente, necessitando também de uma porta paralela e uma interface paralela. A forma de transmissão dos dados em serial e também paralela pode ser dividida em três categorias: (i) **Simplex** - o dado pode ser transmitido em uma única direção, um de cada vez; (ii) **Half duplex** - o dado pode ser transmitido nas duas direções, contudo, um por vez; e (iii) **Full duplex** - o dado pode ser transmitido nas duas direções e simultaneamente. A transmissão dos dados também pode ocorrer de forma síncrona, onde os dados são transmitidos continuamente em intervalos regulares seguindo o clock da CPU. Há também a forma assíncrona onde os dados são enviados em intervalos irregulares (aleatórios). A transmissão assíncrona ocorre pois o transmissor divide a informação em pacotes, os encapsula e transmite separadamente. Para que isso seja possível, é enviado o início e o fim de cada pacote, além disso a taxa de bits transmitidos em um determinado tempo deve ser igual no receptor e no emissor. Um dos protocolo serial I/O mais implementador para transmissão síncrona ou assíncrona é a RS-232 ou EIA-232.

Usualmente, os dispositivos de I/O são os mais lentos dentro de um sistema digital. Eles podem impactar negativamente a performance de todo o sistema e se tornar o gargalo. Por isso, a escolha por qual usar, depende do tipo da aplicação. Aplicações que processam grandes quantidades de dados devem olhar para primeira medida, enquanto se o problema é acesso a arquivos a última deveria ser observada.

O barramento é componente que interliga todos os principais componentes de um sistema embarcado: o processador principal, os componentes de I/O e a memória. O barramento, nada mais é que simples conexões de fios ou caminho/trilhas condutoras que transmitem dados, endereços e sinais de controle (sinal de clock, requisições).

Em sistemas digitais mais complexos, o número de barramentos aumenta e faz-se o uso de *bridges* para conectar vários barramento e carregar informação de um barramento para outro, facilitando assim, que alguns componentes se inter-comuniquem. A bridge pode automaticamente: fornecer um mapeamento transparente da informação de endereço

quando o dado é transferido de barramento para outro, implementar diferentes sinais de controle para vários barramentos, modificar o dado que está sendo transmitido se qualquer protocolo de transferência de um barramento varia de outro.

Os barramentos podem ser divididos em três principais categorias: barramento de sistema, barramentos de I/O e barramento backplane. Os barramento de sistema (barramento principal, local ou processador-memória) conecta a memória principal e a cache ao processador principal. São geralmente curtos, de alta velocidade e barramentos customizados. O barramento backplane são barramentos mais rápidos que conectam a memória, o processador principal e I/O, todos em um só barramento. Por último, o barramento de I/O (barramento de expansão, externo ou host) atua como uma “extensão” do barramento de sistema para conectar os demais componentes ao processador principal. Além disso, faz a conecção entre outros barramentos por meio de bridges, com outros componentes, e/ou conecta o próprio SE com ele mesmo, por meio de comunicação com a porta I/O. A principal diferença entre o barramento de sistema e o de I/O é a possibilidade da presença de sinais de controle de IRQ (request de interrupção) em um barramento I/O.

Para cada categoria descrita, ainda há subdivisões em relação a se o barramento é expansível ou não-expansível. Um barramento expansível é aquele em que componentes adicionais podem ser conectados na placa durante o uso, enquanto o barramento não-expansíveis é aquele em que componentes adicionais não podem ser simplesmente conectados à placa e logo em seguida, se comunicarem por meio deles com os outros componentes. Apesar dos barramentos expansíveis serem mais flexíveis, eles tendem a ser mais custosos.

Cada barramento possui um protocolo relacionado a como um dispositivo deve usar. É ele que define quais dispositivos vão ganhar acesso ao barramento (arbitragem) e quais as regras cada dispositivo deve seguir para comunicar por meio do barramento (handshaking), além dos sinais relacionados a cada barramento. Dependendo do barramento, é possível que um árbitro de barramento conceda a um mestre a opção de realizar toda a sua transmissão (atômico) ou que a sua transmissão seja dividida (*split*) entre vários mestres. As regras que os dispositivos devem seguir (*handshake*) é dependente de cada barramento e a forma como os dispositivos comunicam. A base de qualquer handshake de um barramento é determinado pelo esquema de tempo de barramento, uma vez que eles são baseados em uma ou na combinação de barramentos síncronos ou assíncronos.

Os dispositivos obtêm acesso a um barramento por meio do esquema arbitragem de barramento. Este é baseado em dispositivos classificado em dispositivos mestres (podem começar uma transação) ou dispositivos escravos (dispositivos que só podem ganhar acesso a um barramento em resposta a requisição do dispositivo mestre). O esquema mais simples é onde o processador principal é o dispositivo mestre e o restante são os escravos. Dessa forma não é necessário nenhum processo de arbitragem. Quando múltiplos mestres são permitidos, torna-se necessário um árbitro (circuito de hardware separado). Ele define, considerando algumas circunstâncias, qual mestre ganha controle sobre o barramento. Há alguns esquemas de bus arbitration, sendo os mais comuns: central dinâmica paralela, serial centralizada (daisy-chain) e auto-seleção distribuída.

A arbitragem central dinâmica paralela é um esquema no qual o árbitro está localizado no centro e todos os mestres estão conectados a ele. Nesse esquema, os mestres recebem acesso ao barramento por meio de um sistema de fila FIFO (first in, first out) ou baseado em prioridade (cada dispositivo mestre ganha uma prioridade). Uma desvantagem é a possibilidade de o árbitro não intervir se um único mestre na frente da fila mantiver o controle do barramento, nunca completando e não permitindo que outros mestres acessem o barramento.

A arbitragem central serializada, também conhecida como arbitragem em cadeia, é um esquema no qual o árbitro está conectado a todos os mestres e os mestres estão conectados em série. Independentemente de qual mestre faz a solicitação para o barramento, o primeiro mestre da cadeia recebe o barramento e passa a “concessão de barramento” para o próximo mestre na cadeia quando o barramento não é mais necessário.

Existem também esquemas de arbitragem de auto-seleção distribuídos, o que significa que não há um árbitro central e nenhum circuito adicional. Nesses esquemas, os mestres se arbitram trocando informações de prioridade para determinar se um mestre de maior prioridade está fazendo uma solicitação para o barramento, ou mesmo removendo todas as linhas de arbitragem e esperando para ver se há uma colisão no barramento. Se houver colisão no barramento significa que mais de um mestre está tentando usar o barramento simultaneamente.

Há somente dois tipos de transação que um dispositivo de barramento pode fazer: receber (*READ*) e transmitir (*WRITE*). Essas transações podem ser feitas entre dois processadores (por exemplo, processador principal e controlador de I/O) ou processador e memória (processador principal e memória).

Um barramento síncrono inclui o uso de um sinal de clock, além dos sinais a serem transmitidos (os dados, endereço e outras informações de controle), isso faz com que tanto o dispositivo que recebe, quanto o que envia, quanto o barramento, tenham a mesma taxa de clock. Além disso, é necessário que ambos usem o mesmo “detector de borda” (borda ascendente ou borda descendente) do ciclo de clock. Tipicamente, barramentos mais rápidos usam esquema de barramento síncrono.

O barramento assíncrono não transmite o sinal de clock. Ele transmite outros sinais de handshake diferentes do clock. Os barramentos que usam o esquema síncrono têm problemas com o tamanho do barramento, uma vez que quanto maior, mais tempo demora para a informação chegar ao seu destino, o que atrapalha no sinal de clock enviado. Já o assíncrono não possui esse problema, uma vez que não usa o sinal de clock, contudo, ele traz uma complexidade maior para os dispositivos coordenarem os comando de requisição, responder a comandos, entre outros. Os dois protocolos mais básicos para o handshaking em barramentos assíncronos é o do mestre indicando ou requisitando uma transação (READ/WRITE) e o escravo respondendo à transição indicando ou requisitando (ACK - confirmação, ou ENQ - consulta).

Atualmente, todos os sistemas utilizam esses quatro componentes. A diferença entre os diversos sistemas está na complexidade dos próprios componentes e como eles se comunicam. Por exemplo, os sistemas embarcados tendem a ter os recursos reduzidos,

todavia com alguns recursos extras como memória RAM, ROM dispositivos de I/O (portas paralelas, seriais e conversores analógico-digitais), tudo em único circuito. Já o caso de desktops, há os microprocessadores, que por si só já possuem internamente todos os componentes, contudo com a RAM e outros componentes externos ao chip. Esses componentes influenciam fortemente a performance do sistema. O projeto de qualquer sistema envolve a escolha desses componentes.

Disponível em: https://github.com/plsilva/Files_DigitalSystems