A Arquitetura ARM

MC722 - Projeto de Sistemas Computacionais

Pedro Henrique Gomes
RA 024844

Tatiane Silvia Leite
RA 025217

Uirauna Imirim Caetano
RA 025336

1. INTRODUÇÃO

A arquitetura ARM (Advanced RISC Machines) começou como um projeto em 1983 na Arcon Computers de Cambridge, Inglaterra, para desenvolver um processador que fosse similar ao já usado MOS Technology 6502.

O desenvolvimento da primeira versão foi terminado em 1985, chamado ARM1. Esta arquitetura não chegou ao mercado e a Arcon continuou no desenvolvimento, terminando no ano seguinte a segunda versão, chamada de ARM2.

Além do 6502, a arquitetura ARM tem uma grande herança do processador Berkeley RISC1, como instruções de tamanho fixo (32 bits), formato das instruções e a arquitetura load-store. Apesar disso, a Arcon optou por ter instruções multi-ciclo, ao contrario do RISC1.

Em sua época, o ARM2 era o processador de 32 bits mais simples no mercado, com apenas 30.000 transistores, mas ainda assim superava o desempenho de muitos processadores (RISC ou CISC) mais complexos, como o 286 da Intel.

O desenvolvimento continuou, e o nome original (Arcon RISC Machine) deixado de lado quando a ARM Ltda. foi criada e assumiu as patentes e o desenvolvimento da arquitetura ARM. Atualmente a arquitetura já conta com sua 11ª versão, mas as versões antigas ainda são usadas e desenvolvidas, já que o uso de cada família é voltado para um nicho de mercado.

Existem também várias extensões especializadas para alguma função ou processamento, como o Jazelle e o Thumb.

1.1 Principais características da arquitetura:

- Processador de 32 bits;
- 16 registradores de uso geral;
- Conjunto de instruções extensível com o uso de co-processadores;
- Instruções básicas similares ao 6502;
- Instruções de três endereços;
- Capacidade de executar instruções de 16 bits usando a arquitetura Thumb;
- Baixo consumo de energia;
- Tamanho do núcleo reduzido;
- Até 16 co-processadores lógicos.

1.2 Tipos de Núcleos:

- Processadores para aplicativos;
- Processadores para sistemas embarcados;
- Processadores SecurCore;

1.3 Famílias:

- ARM7 Thumb;
- ARM9 Thumb;
- ARM9E;
- ARM10E;
- ARM11;
- SecurCore;
- OptimoDE Data Engine;
- Cortex Family;

2. CONJUNTO DE INSTRUÇÕES DA ARQUITETURA ARM

A arquitetura ARM foi desenvolvida para possibilitar implementações muito enxutas, sem deixar de lado o alto desempenho. Isso é possível pela simplicidade dos processadores ARM. Importante lembrar que implementações pequenas implicam em baixo consumo de energia, o que torna esses processadores interessantes para aplicações móveis.

O ARM é tipicamente um RISC (Reduced Instruction Set Computer). Algumas características interessantes das instruções ARM:

- Conjunto grande e uniforme de registradores;

- Arquitetura de LOAD / STORE. Operações de processamento de dados não operam diretamente com o conteúdo da memória, somente com o conteúdo de registradores;
- Modos de endereçamento simples, com todos endereços de load / store sendo determinados a partir dos registradores ou pelos campos da instrução;
- Uniformidade e tamanho fixo dos campos das instruções para simplificar a decodificação de instruções;
- Controle sobre a ALU e sobre o shifter (deslocador) em todas instruções de processamento de dados;
- Auto incremento e decremento dos endereços das instruções;
- Instruções de múltiplos loads / stores para maximizar a performance;
- Execução condicional da maioria das instruções.

2.1 Registradores

A arquitetura possui 31 registradores de propósito geral, todos de 32 bits. Em qualquer momento apenas 16, dos 31, registradores são visíveis. Os registradores restantes são usados em operações de exceção, quando o processador entra em um de seus modos especiais de operação e substitui alguns dos 16 registradores comuns por registradores específicos do modo.

Dos 16 registradores, 2 têm papel especial:

- Link Register (R14) - possui o endereço da próxima instrução a ser executada após um Branch and Link (BL), que é uma instrução usada na chamada de subrotinas. Excluindo essa situação, o R14 pode ser utilizado normalmente pelo programador.
- Program Counter (R15) - possui o endereço da próxima instrução a ser executada pelo processador. Sempre possui o valor do endereço da instrução atual mais 8 bytes.

É comum o uso do registrador R13 para fins de Stack Pointer; todos outros 13 podem ser usados para propósito geral. A partir da versão 3 o ARM passou a separar em registradores independentes o seu status. Existem dois registradores responsáveis por essa função: o Current Program Status Register (CPSR) e o Saved Program Status Register (SPSR), ambos de 32 bits.

O primeiro possui 4 bits condicionais de flag (Negative, Zero, Carry e Overflow), 2 bits de controle de interrupção, 5 para controle do modo atual do processador e 1 que indica o tipo de instrução que está sendo executada (ARM ou Thumb). O segundo registrador de status guarda o conteúdo do primeiro durante uma exceção, para possibilitar um retorno seguro do contexto anterior da CPU; existe um registrador SPSR para cada modo de exceção do processador. A figura 1 mostra a disposição dos 31 registradores e os modos de operação que podem acessá-los.

2.2 Modos da Arquitetura ARM

A arquitetura ARM suporta até 7 modos de operações, apresentados a seguir.

- User: execução normal de programas. Possui restrições de acesso a registradores;
- FIQ (Fast Interrupt): suporta a transferência rápida de dados;
- IRQ (Interrupt): usado para manipulação de interrupções de propósito geral;
- Supervisor: é um modo protegido para o sistema operacional;
- Abort: implementa memória virtual e/ou proteção de memória;
- Undefined: suporta emulação em software de co-pro-cessadores;
- System: executa tarefas privilegiadas do sistema operacional, existente a partir da versão 4 do ARM.

Os modos de operação podem mudar através de controle de software ou através de interrupções externas. Durante a execução normal de um programa o processo encontra-se no modo User.

2.3 Tipos de Instruções

O conjunto de instruções do ARM pode ser dividido em 6 grandes classes:

- Instruções de Branch;
- Instruções de processamento de dados;
- Transferência de registradores de status;
- Instruções de Load / Store;
- Instruções de co-processador;
- Instruções de geração de exceções.

Uma característica muito importante das instruções ARM diz respeito aos bits de condições. A maioria das instruções de processamento pode atualizar os 4 bits de flag contidos no registrador CPSR, que indicam a ocorrência de uma resultado nulo (Zero), de resultado negativo (Negative), de Carry ou Overflow. Quase todas instruções ARM possuem 4 bits condicionais que especificam se a instrução será executada ou não, a partir da situação em que os bits de flag se encontram. Com esses 4 bits condicionais é possível existir até 16 situações condicionais; dessas, uma é utilizada para execução incondicional (sempre) de uma instrução a outra para instruções que não possibilitam execução condicional.

Das outras 14 situações condicionais que restam podem ser testadas condições tais quais:

- Igualdade e desigualdade;
- Menor que, menor ou igual que, maior que, maior ou igual que, em aritmética sinalizada e não-sinalizada;
- Situação de cada flag do registrador CPSR individualmente.

Quando o processador encontra uma instrução com campo condicional válido, ele compara a condição desejada com o status dos flags do CPSR; se ambos concordam a instrução é executada normalmente, caso contrário não. Os

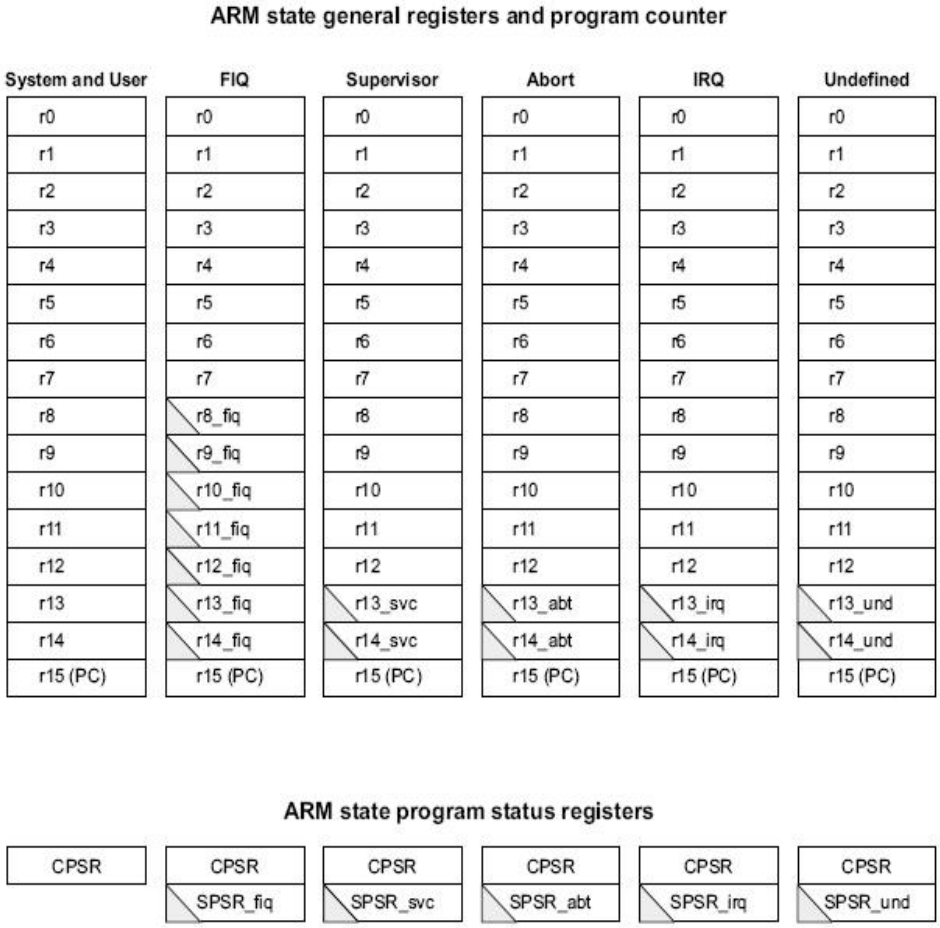

Figure 1: Tabela de Registradores

Condition Field		
Mnemonic	**Description**	**Description (VFP)**
EQ	Equal	Equal
NE	Not equal	Not equal, or unordered
CS / HS	Carry Set / Unsigned higher or same	Greater than or equal, or unordered
CC / LO	Carry Clear / Unsigned lower	Less than
MI	Negative	Less than
PL	Positive or zero	Greater than or equal, or unordered
VS	Overflow	Unordered (at least one NaN operand)
VC	No overflow	Not unordered
HI	Unsigned higher	Greater than, or unordered
LS	Unsigned lower or same	Less than or equal
GE	Signed greater than or equal	Greater than or equal
LT	Signed less than	Less than, or unordered
GT	Signed greater than	Greater than
LE	Signed less than or equal	Less than or equal, or unordered
AL	Always (normally omitted)	Always (normally omitted)

Figure 2: Sufixos Condicionais

bits condicionais são ativado através de sufixos nas instruções, conforme é mostrado na figura 2. Exemplificando, o sufixo EQ indica que a instrução só é executada se o bit Z (Zero) estiver ativado; assim, a instrução ADDEQ R1,R2, 20 somente adicionará 20 ao registrador R2 e armazenará em R1 se a operação anterior tiver ativado o bit Z.

2.3.1 Instruções de Branch

Essas instruções possibilitam a troca do fluxo de instruções escrevendo um endereço no PC. As instruções Branch comuns possuem um offset sinalizado de 24 bits, possibilitando desvios de 32 MB. Existem ainda outros 3 tipos de instruções Branchs. O primeiro, Branch and Link preserva o endereço anterior no registrador R14, podendo ser utilizado em chamadas de subrotinas. Um segundo tipo realiza a troca do tipo de instruções, de ARM para Thumb, ou vice-versa. Por último, existe um tipo de branch que troca para o modo de execução de bytecodes Java (extensão Jazelle).

2.3.2 Instruções de Processamento de Dados

Existem 12 instruções que têm o mesmo formato e fazem operações lógicas e aritméticas com até dois operandos de origem, escrevendo o resultado em um terceiro operando, de destino. Essas instruções podem atualizar os bits de flag. Como origem podemos ter registradores, imediatos ou registradores deslocados.

Há ainda 4 tipos de instruções de comparação, que usam o mesmo formato que as instruções lógico-aritméticas. Essas instruções, porém, não escrevem o resultado em um registrador de destino, mas sempre atualizam os bits de flag.

Instruções de multiplicação operam sobre dois registradores de 32 bits. O resultado pode ser de dois tipos: de 32 bits, que é escrito em um único registrador, e de 64 bits, que é escrito em dois registradores.

2.3.3 Transferência de Registradores de Status

Existem duas instruções (MRS e MSR) que podem transferir dados do registrador de status. A primeira move o conteúdo do registrador de status para um registrador de propósito geral e a segunda faz o inverso. A instrução MSR pode ser usada para ajustar os valores dos flags, valores dos bits de interrupção ou para mudar o modo do processador.

2.3.4 Instruções de Load/Store

Essas instruções podem ser de 3 tipos que fazem load / store de 1 único registrador, fazem load / store de vários registradores ou que trocam o conteúdo de um registrador com um endereço da memória.

Instruções de um único registrador acessam words, half-words e bytes. Como podem acessar o PC, são usadas para jumps que alcançam os 4 GB de endereçamento.

As instruções LDM e STM, respectivamente, fazem load e store de vários registradores ao mesmo tempo. Podem ser usadas no início e fim de subrotinas para empilhar os registradores que devem ser preservados.

A instrução SWP faz a troca do conteúdo de um registrador com um endereço de memória.

2.3.5 Instruções dos co-Processadores

São instruções que podem iniciar uma operação no co-processador, podem também transferir dados deste último para a memória e vice-versa, e ainda podem ser usadas para transferir dados entre registradores do ARM e do seu co-processador.

2.3.6 Instruções de Geração de Exceções

A instrução SWI causa uma interrupção de software, sendo normalmente usada para fazer chamadas ao sistema operacional. A instrução BKPT causa uma exceção de aborto. Caso uma rotina de tratamento esteja instalada no vetor de interrupções essa exceção é tratada como um breakpoint e se há um hardware de debug instalado ele também pode tratar essa instrução como um breakpoint.

2.4 Código ARM

A característica mais importante da arquitetura ARM é a grande densidade de seu código. Podemos analisar isso através do exemplo do algoritmo euclidiano de MDC (Máximo Divisor Comum) apresentado a seguir.

- Código em C:

```
int mdc (int i, int j) {
   while (i != j)
      if (i > j)
         i -= j;
      else
         j -= i;
   return i;
}
```

- Código em ARM Assembler:

```
b test
loop subgt R0,R0,R1
suble R1,R1,R0
test cmp R0,R1
bne loop
```

3. SISTEMA DE MEMÓRIA

Os processadores ARM podem ser encontrados nas mais diversas aplicações, e por essa razão, os requisitos de memória variam, utilizando-se dos seguintes recursos: múltiplos tipos de memória, caches, buffers de escrita, memória virtual e outras técnicas de mapeamento. O CP15 (co-processador), também conhecido como co-processador de Controle de Sistema, é responsável por realizar o controle do sistema de memória padrão do ARM e suas facilidades. Ele pode conter até 16 registradores primários, com 32 bits cada um. E para alguns destes registradores, existem bits adicionais usados para identificar uma versão específica do registrador e/ou o tipo de acesso específico do registrador.

3.1 Mais Detalhes sobre o Sistemas de Memória

Os processadores ARM (e softwares) foram projetados para serem conectados a uma memória endereçada a byte, ou seja, cada endereço da memória corresponde a um byte. Então para ter acesso a uma *word* os dois bits menos significativos são ignorados, e caso se queira ter acesso a uma *halfword* apenas o bit menos significativo é ignorado. O formato dos dados (Little Endian ou Big Endian) de um processador ARM é igual ao formato do sistema

de memória. A memória que é usada para armazenar os programas está dividida em: Memória Principal (RAM) e Memória ROM. O sistema de memória de primeiro nível é composto por cache de instrução e dados separados, áreas separadas de memória para instruções e dados, sistema DMA para acessar a memória, buffer de escrita e duas micro-TLBs.

3.1.1 Cache

Cada local de memória na cache é conhecido como linha de cache. A cache no ARM é f*ull associative*, o que signifca que ela é associativa por conjunto de N vias, sendo N o número total de linhas de cache, assim, qualquer busca na cache precisa checar cada linha de cache. A cache é também mapeada diretamente, ou seja, a cache é associativa por conjunto de m vias, com um conjunto de m blocos e uma entrada pode residir em qualquer bloco dentro desse conjunto. O tamanho das linhas de cache é sempre uma potência de dois e tipicamente são de 16 bytes (4 words) ou de 32 bytes (8 words). A política de substituição utilizada pela cache dos processadores ARM podem ser: peseudo-rândomica ou Round-Robin. Quanto ao tamanho, o sistema de memória ARM permite diferentes tamanhos de cache, dependendo da importância de performance no dispositivo. Os processadores ARM dividem sua cache em duas partes: uma cache exclusiva para dados e uma cache exclusiva para instruções, podendo assim acelerar o processo de execução já que é permitido instruções de leitura e escrita num mesmo ciclo de clock. E o tamanho da cahe de dados pode diferir do tamanho da cache de instrução. Para cada cache, existe uma área da memória RAM dedicada também separadamente, uma para dados e outra para instruções, essas áreas são chamadas TCM. A TCM foi desenvolvida com objetivo de fornecer uma memória mais estável que as caches e ao mesmo tempo com baixa latência. Assim, elas são utilizadas para conter rotinas críticas, em que a imprevisibilidade da memória cache é altamente indesejável, como por exemplo: rotinas manipuladoras de interrupções.

3.1.2 Buffer de escrita

É um bloco de memória de alta velocidade que tem como objetivo otimizar as gravações na memória principal. Quando deve ocorrer a gravação, os dados, o endereço e outros detalhes são escritos no buffer de escrita (com alta velocidade). E o buffer de escrita fica responsável em completar a gravação na memória principal, na velocidade da memória principal, que é tipicamente menor que a velocidade do processador ARM.

3.1.3 Interrupções

As interrupções nos processadores ARM são sinalizadas externamente e muitas implementações sincronizam as interrupções antes de gerar uma exceção. Elas podem ser:

- FIQ :solicitação de interrupção rápida;
- IRQ: solicitação de interrupção normal.

4. SISTEMA DE GERENCIAMENTO DE MEMÓRIA

A MMU controla o acesso à memória externa e é responsável pela tradução do endereço virtual em endereço

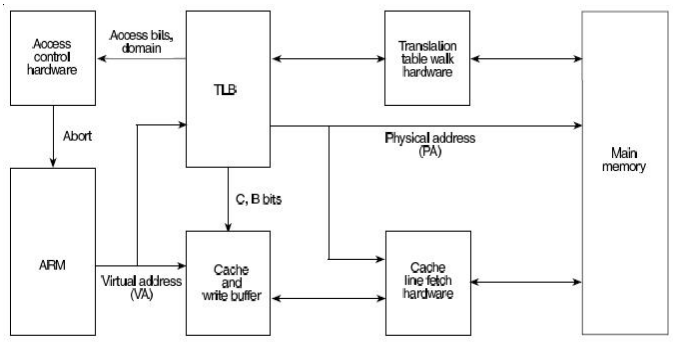

Figure 3: Sequência de Acesso à Memória

físico, assim como a checagem de permissão de acesso. Através das tabelas de tradução armazenadas na memória é obtido quase todo controle detalhado do sistema, e as entradas dessas tabelas é que definem as propriedades de áreas de memória que variam de 1KB a 1MB de tamanho. Essas propriedades incluem:

- Mapeamento de endereço virtual para físico (este endereço físico identifica que localização da memória principal está sendo acessada);
- Permissões de aceso a memória (quando um acesso não é permitido, um sinal de *memory abort* é enviado para o processador);
- Bits de cachability e bufferability;
- Registradores do CP15 que permitem controle de alto nível, como a localização de tabelas de tradução. Também são utilizados para fornecer a situação (status) dos *memory aborts* para o ARM;

Visando reduzir o custo médio de acesso a memória, os resultados das buscas nas tabelas de tradução são armazenadas em estruturas de cache, também chamadas de TLBs (Translation Lookaside Buffers). Geralmente, um sistema que possui apenas uma interfaze de memória possui uma única TLB, já um sistema que possui interface de memória de instrução e interface de memória de dados separadas, normalmente possui TLBs separadas também, uma para instrução e uma para dados.

4.1 Seqüencia de Acesso à Memória

Ao ser gerado um acesso a memória pelo processador ARM, a MMU procura o endereço virtual de acesso na TLB (ou nas TLBs). Caso a TLB não contenha a entrada para o endereço virtual, o hardware é invocado, retornando a tradução e a permissão de acesso da tabela de tradução na memória principal. A informação retornada é colocada na TLB em uma entrada não utilizada, ou sobrescrevendo alguma entrada existente.

A MMU pode ser habilitada ou desabilitada, para isso basta alterar o bit menos significativo do registrador 11 do CP15. Ao desabilitar a MMU, os acessos a memória passam a ser tratados da seguinte forma:

- A aplicação é que determina se o uso de caches e buffers de escrita serão habilitados ou não;

- Não são realizadas verificações de permissão de acesso a memória, e não são mais gerados sinais de abortos pela mmu;
- O endereço físico é igual ao endereço virtual, para todos os acessos.

Ao habilitar ou desabilitar a MMU altera-se o mapeamento de endereço virtual-para-físico, então podem ocorrer alguns problemas caso o endereço físico do código que habilita ou desabilita a MMU não seja igual ao endereço virtual, devido ao *prefetch* de instrução.

4.2 Processo de tradução

A MMU suporta acessos de memória baseados em seções ou páginas. As seções compreendem blocos de memória de 1MB. Já as páginas podem ter três tamanhos diferentes:

- Minúsculas: compreendem blocos de memória de 1KB;
- Pequenas: compreendem blocos de memória de 4 KB (com o controle de acesso dividido em subpáginas de 1KB);
- Grandes: compreendem blocos de memória de 64 KB (com o controle de acesso dividido em subpáginas de 16KB).

Uma coleção de seções, de páginas grandes e de páginas pequenas é chamada de domínio. O acesso a cada domínio é controlado pelo registrador 3 do CP15, sendo que cada campo permite habilitar ou desabilitar o acesso ao domínio inteiro rapidamente.Assim, o mecanismo de troca de contexto entre processos é bastante eficiente, já que áreas completas de memória podem ser colocadas e retiradas da memória virtual rapidamente.

A tabela de tradução mantida em memória é dividida em dois níveis:

1. Tabela de primeiro nível: armazena traduções de seções e ponteiros para tabelas do segundo nível;
2. Tabela de segundo nível: armazena traduções de pá-ginas pequenas e grandes. Existe um tipo desse nível que pode armazenar a tradução de páginas minúsculas.

O processo de tradução sempre começa do mesmo jeito (com um fetch no primeiro nível) em seqüencia existem quatro formas de realizar a tradução de endereços e a verificação de permissão. Estas maneiras dependem se o endereço está marcado como um acesso mapeado a seção (que requer somente um fetch no primeiro nível) ou a página (que requer um fetch no nível segundo também), sendo que se for mapeado a página, pode ser página grande, pequena ou minúscula.

4.3 Aborts

Restrições de memória podem parar a execução do ARM, e podem ser: falha da MMU (MMU detecta a restrição e sinaliza para o processador) e aborto externo (o sistema externo de memória sinaliza um acesso ilegal à memória). As falhas geradas pela MMU podem ser falha de alinhamento, falha de tradução, falha de domínio e falha de permissão. E pode abortar 3 tipos de acesso: *fetch* de linha, acessos a memória e acessos a tabela de tradução. Registradores específicos para falhas:

1. FAR (Fault Adress Register): onde é escrito o endereço que causou a falha;
2. FSR (Fault Status Register): é atualizado com status de falha.

A arquitetura ARM também define um pino para abortos externos, permitindo que os acessos de leitura, escritas sem buffer, descritor de fetch de primeiro nível, descritor de fetch de segundo nível e semáforos em áreas de memória possam ser abortados externamente e reiniciados de forma segura.

4.4 Registradores CP15

Os registradores 1 (alguns bits),2,3,4,5,6,8 e 10 do CP15 controlam a MMU, abaixo será mostrado uma breve descrição desses registradores.

- Registrador 1: habilita/desabilita MMU;
- Registrador 2: fornece o endereço físico da tabela de tradução de primeiro nível atualmente ativa a partir do valor dos 18 bits mais significativos;
- Registrador 3: cada dois bits definem a permissão de acesso para cada domínio (coleção de seções, paginas grandes e pequenas).
- Registrador 4: reservado;
- Registrador 5: status de falha (FSR);
- Registrador 6: endereço de falha (FAR);
- Registrador 8: utilizado para controlar TLBs;
- Registrador 10: permite que os resultados de uma busca nas tabelas de tradução sejam carregados na TLB de uma forma que não sejam sobrescritos pelos resultados de buscas posteriores.

5. PIPELINE

O pipeline dos processadores baseados na arquitetura ARM são muito semelhantes aos pipelines de outros processadores RISC, como o MIPS. Como atualmente existem varias famílias de processadores ARM, cada uma usando um diferente modelo de pipeline, vamos focar no pipeline do Intel XScale, um processador para aplicativos de alto desempenho baseado no ARMv5TE e desenvolvido a partir da arquitetura StrongARM.

As principais diferenças entre o pipeline do XScale e dos processadores da arquitetura StrongARM são o número de estágios (cinco no StrongARM e sete no pipeline principal do XScale) operando em uma freqüência mais alta e a existência de dois módulos secundários paralelos ao módulo principal, como pode ser visto na figura 4. Um pipeline mais longo como o usado no Xscale traz algumas desvantagens relacionadas a penalidades de erros no branch-prediction e uso de loads, mas no geral as outras medidas tomadas causam um ganho de desempenho significativo.

Esse ganho de desempenho vem da maior freqüência do pipeline e também da capacidade de processamento paralelo obtida com o uso de módulos secundários. O módulo principal, também chamado de Main Execution Pipeline ou ULA, é responsável pelo processamento de

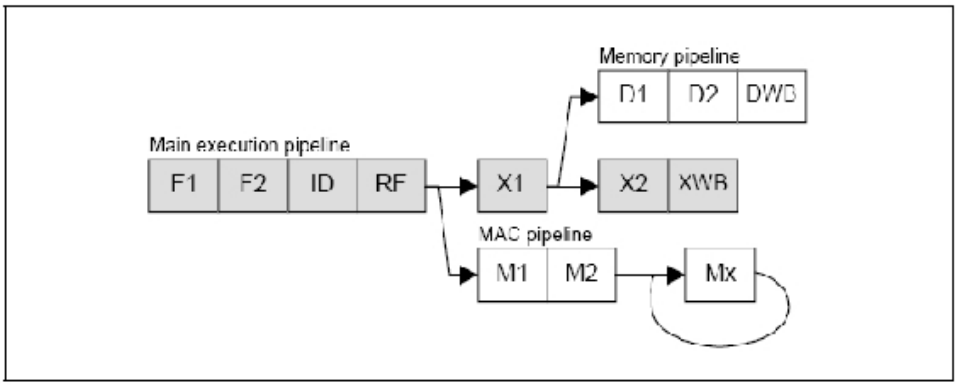

Figure 4: Fluxo do Pipeline

Pipelines and Pipe stages

Pipe / Pipestage	Description
Main Execution Pipeline	Handles data processing instructions
IF1/IF2	Instruction Fetch
ID	Instruction Decode
RF	Register File / Operand Shifter
X1	ALU Execute
X2	State Execute
XWB	Write-back
Memory Pipeline	Handles load/store instructions
D1/D2	Data Cache Access
DWB	Data cache writeback
MAC Pipeline	Handles all multiply instructions
M1-M5	Multiplier stages
MWB (not shown)	MAC write-back - may occur during M2-M5

Figure 5: Estágios do Pipeline

instruções lógico-aritméticas assim como outras instruções mais simples do processador, como branch, enquanto o módulo de memória, também chamado Memory Pipeline, é responsável pelo processamento de instruções relacionadas ao uso da memória e o módulo de multiplicação, também chamado de MAC Pipeline, é responsável pelas operações de acumulação e multiplicação.

- Main Execution Pipeline: Este pipeline é responsável pela leitura (F1 e F2) e decodificação (ID) das instruções e pela leitura dos registradores (RF). Após a leitura dos registradores, se a instrução envolver operações de multiplicação ou acumulação, ela será enviada então para o MAC que continua seu processamento. Caso contrário, a instrução continua no pipeline principal e passa pela etapa de execução na ALU (X1) e novamente, dependendo da instrução, pode ser desviada para o pipeline de memória. Caso contrário, ele continua no pipeline principal e termina o processamento nas duas fases seguintes.

- MAC Pipeline: O pipeline responsável pelas instruções de multiplicação merece uma atenção especial pelo fato de não ser exatamente um pipeline, já que apenas uma instrução por vez pode ser processada. Isso se deve ao fato de que o MAC não tem comprimento fixo, e a instrução pode percorrer varias vezes um mesmo estágio. O número de ciclos de clock necessários para completar uma instrução no MAC é definido pelo tipo e argumentos da instrução. Para realizar as multiplicações o MAC usa um registrador de 40 bits chamado acc0, que pode no fim da instrução ser copiado para os registradores de uso geral do núcleo.

- Memory Pipeline: Apesar de ser o mais comprido, este é o pipeline mais simples do Xscale. Ela é dividida em duas partes: os estágios D1 e D2 e o estágio DWB. Nos estágio D1 e D2 as instruções load/store são processadas e os dados são obtidos da cache, e depois no estágio DWB os dados são escritos na cache.

6. EXTENSÕES DA ARQUITETURA

6.1 Thumb

As instruções Thumb são na verdade instruções ARM codificadas para aumentar a performance do processador em certas aplicações, utilizando um barramento de dados mais simples (de apenas 16 bits) e criando uma melhor densidade dos códigos. As versões de processadores ARM que possuem o conjunto de instruções Thumb têm a letra T em seu nome, como ARM720T. O uso de instruções Thumb é recomendado em aplicações que não utilizam toda largura do barramento de memória, como algumas aplicações embarcadas.

Todas instruções Thumb são codificadas em 16 bits. As operações matemáticas continuam sendo de 32 bits, somente o tamanho da instrução é que passa a ser de 16 bits. Quando o processador está em modo Thumb somente 8 registradores (R0 - R7) estão disponíveis para uso, além do PC (R15) e do Link Register (R14). O uso do modo Thumb é especificado através de um bit (bit T) do registrador CPSR. Quando o bit T está ativado a fase de fetch do processador busca instruções Thumb, de 16 bits.

6.2 Thumb-2

Esse tipo novo de instrução foi implementado inicialmente no núcleo ARM1156 (em 2003). Ele extende o limite de 16 bits do modo Thumb com algumas instruções adicionais de 32 bits, para dar um melhor fôlego para aplicações totalmente escritas em modo Thumb. Assim, essas aplicações podem continuar com a densidade de código do antigo Thumb, mas tem a possibilidade de utilizar recursos do modo ARM.

6.3 Jazelle

Essa tecnologia permite a execução nativa em hardware de bytecodes Java. É um outro modo de operação implementado na arquitetura, diferente dos modos ARM e Thumb. Os processadores que suportam o modo Jazelle têm o seu nome acompanhado da letra J, como por exemplo, ARM926J-S.

6.4 Thumb-2EE

Também conhecido como Jazelle RCT, essa tecnologia foi lançada nesse ano, primeiramente no núcleo Cortex-A8. O Thumb-2EE extende o Thumb-2 com um conjunto de instruções particularmente úteis para códigos gerados em tempo de execução, como ocorre com os compiladores JIT do Java e C#. Outras linguagens que podem obter vantagens da tecnologia são Perl e Phyton. Com o Thumb-2EE códigos mais compactos podem ser criados, sem que isso impacte na performance da aplicação.

6.5 NEON

É um conjunto de instruções do tipo SIMD (Single Instruction Multiple Data) de 64 e 128 bits criado especialmente para acelerar aplicações multimidia e de processamento de sinais. A tecnologia possui um novo conjunto de instruções, um conjunto separado de registradores e um hardware de execução independente. As instruções suportam dados de 8, 16, 32 e 64 bits, inteiros e de ponto flutuante de precisão simples, operando em modo SIMD. Com o NEON é possível executar um decoder de MP3 com menos de 10 Mhz e um codec GSM AMR com apenas 13 Mhz. Até 16 operações simultâneas são suportadas pelas instruções SIMD criadas.

6.6 VFP - Vector Floating Point Processor

É um coprocessador que extende as instruções ARM para o cálculo de ponto flutuante de precisão simples e dupla, totalmente compatível com o padrão ANSI/IEEE 754. As aplicações que fazem uso do co-processador são: compressão e descompressão de voz, gráficos 3D, audio digital, PDAs, smartphones, dentre outras. A execução de operações sobre pequenos vetores é privilegiada com o VFP, já que podem ser realizadas operações paralelas

7. APLICAÇÕES

Os processadores ARM representam atualmente mais de 75% do mercado de processadores 32 bits. A ARM oferece a especificação de diversos modelos de processadores com enfoque em alguns tipos específicos de aplicações, que vão desde aplicações móveis e de processamento de imagem, até aplicações automotivas e industriais e na área de segurança. Os processadores ARM são divididos em 3 principais categorias. Os principais modelos de cada categoria estão citados na introdução.

A ARM produz apenas as especificações dos processadores, deixando que outras empresas os fabriquem. Isso torna um pouco difícil descobrir se um determinado produto possui um processador ARM. Ainda, grande parte desses fabricantes integram outros dispositivos, principalmente co-processadores específicos, ao processador ARM, criando maior versatilidade ao projeto de seu produto. Abaixo temos uma lista de alguns produtos que utilizam processadores ARM.

- Gameboy Advance games console;
- Datcom 2000 digital satellite receiver;
- LG Java computer;
- Lexmark Z12/22/32/42/52 Jetprinter;
- Vários handsets GSM da Alcatel, AEG, Ericsson, Kenwood, NEC e Nokia;
- Cable/ADSL modems, da Caymen Systems, D-Link e Zoom;
- 3Com 3CD990-TX-97 10/100 PCI NIC;
  item HP/Ericsson/Compaq pocket PCs;
- Palm PDAs

8. CONCLUSÃO

A arquitetura ARM é implementada em mais de 75% dos processadores de 32 bits atuais principalmente devido à sua versatilidade de uso em diversas aplicações prática, pela boa performance que apresenta e pela constante atualização de novas versões que são lançadas pela ARM. É curioso notar que uma empresa (ARM) que não produz processadores consegue o domínio de grande fatia do mercado. Esse sucesso pode ser atribuido à dedicação exclusiva da ARM no desenvolvimento do projetos dos processadores, sem se preocupar com questões mercadológicas do produto final; esse papel é desempenhado por empresas com a Intel, que compram o projeto de um processador e passam a fabricá-lo e integrá-lo em seus produtos.

Apesar de muito utilizados, os processadores ARM são muito pouco conhecidos, principalmente pelos usuários finais. Os processadores ARM são principalmente utilizados hoje em dia em dispositivos móveis, como PDAs, smartphones e equipamentos de rede. A grande expansão desse mercado e a adoção por parte da Intel da arquitetura ARM a partir de seus processadores XScale, são fatores que prometem perpetuar por muito tempo o uso dessa arquitetura no dia-a-dia do consumidor.

9. BIBLIOGRAFIA

1. Steve Furber, *ARM System Architecture*. Addison-Wesley. (1996)
2. David Seal, *ARM Architecture Reference Manual*. Addison-Wesley. (2000)
3. *ARM7EJ-S Technical Reference Manual*. ARM Ltd. (2001)
4. *Intel XScale® Core Developer's Manual*. Intel Ltd. (2005)