Aula 7

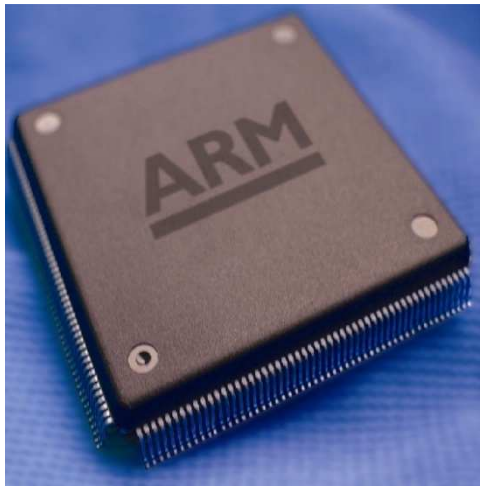
Engenharia de Sistemas Embarcados

Prof. Abel Guilhermino

Tópico: Arquitetura ARM

Arquitetura ARM

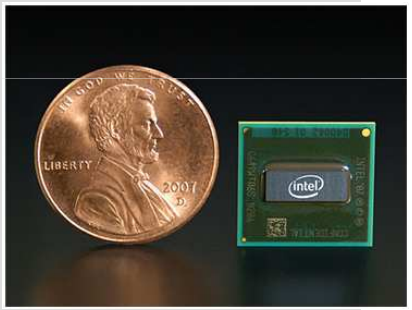
- Advanced RISC Machine (Máquina RISC Avançada)
- São CPUs de 32 bits que utilizam a filosofia RISC.



História do ARM

- Surgiram no início da década de 80, resultantes de um projeto bem-sucedido da **Acorn Computer Group**, Cambridge, Inglaterra.
  - Computadores que usavam microprocessadores de 8 bits (ex: 6502)
- 1985, primeiros exemplares do **ARM1** (Acorn RISC Machine 1)
  - Barramento de Dados 32 bits
- **ARM2 e ARM3**
  - Usados nos primeiros computadores RISC pessoais (Ex: Archimedes 300, 400 e 500) fabricados pela Acorn.
- 1990, Acorn, Apple e a VLSI Technology uniram-se para formar a **Joint Venture ARM Limited**.
- 1991, primeira versão embedded comercial da CPU ARM, o **ARM6**.

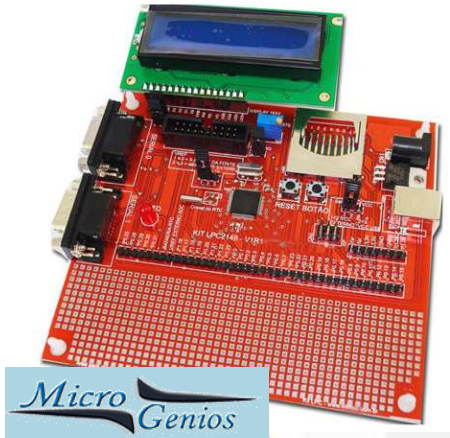
ARM Limited



- Atualmente a ARM Limited não fabrica chips, apenas licencia o uso de suas CPUs (a sua propriedade intelectual) por outros fabricantes de microcontroladores e microprocessadores como (Cirrus Logic, STMicroelectronics, Freescale, OKI, Nitendo, Fujitsu, Intel, VLSI, Sansung Sharp).
- Modelos de Negócio da ARM
  - Licenciamento simples dos núcleos da CPU
    - Fabricante recebe netlist + materiais necessários aos testes e integração do chip
  - Licenciamento dos núcleos de CPU em código sintetizável
    - Além de tudo listado acima, o código HDL.

Características

- Tornou-se líder de mercado para aplicações embarcadas de baixa potência.
- O processador ARM é um processador RISC
- ARM foi o primeiro microprocessador RISC desenvolvido para uso comercial
- A combinação de um hardware simples com um repertório de instruções reduzido permite eficiência no consumo de potência e tamanho reduzido.



Versões da CPU ARM

- V1: primeira versão; incluía um conjunto de 16 registradores da CPU, instruções básicas de carga e armazenamento; suportando dados de 8, 32 ou múltiplos de 32 bits. Não foi usada em **nenhum produto comercial**.
- V2: incluía **instruções de multiplicação**, MAC, suporte a co-processador; bancos de registradores para interrupções rápidas (FIQ).
- V3: Expandiu o barramento de endereços de 32 bits (permitindo endereçar até 4GiB); novos registradores de estado da CPU; conjunto alternativo de instruções de 16 bits (**Thumb**)=> aumenta densidade de código com um mínimo de impacto na velocidade de execução.

Versões da CPU ARM

- V4: adicionou instruções de manipulações de 16bits
- V5: adicionou novas instruções (ex: contagem de zeros a esquerda, breakpoint por software, etc). Melhorou a interoperabilidade entre os modos ARM e Thumb. Incluiu 2 módulos de suporte a CPU: instruções avançadas de Processamemto digital de sinal (v5TE) e o módulo de acelerador Java (Jazelle)(v5TEJ).
  - Jazelle é uma máquina virtual Java multitarefa implementada em hardware, que permite que Java Bytecode seja executado diretamente na arquitetura do ARM como um terceiro estado de execução ao longo dos existentes (ARM e Thumb) = Máquina Virtual Java.
- V6: Suporte a memória; multiprocessamento e gerenciamento de exceções. Inclui os módulos DSP e Java da versão 5. Inserida instruções especiais para processamento de audio e vídeo. Introduziu-se também a versão Thumb-2 (aumento da funcionalidade do conjunto de instruções Thumb de 16bits, incluindo instruções de 32 bits).

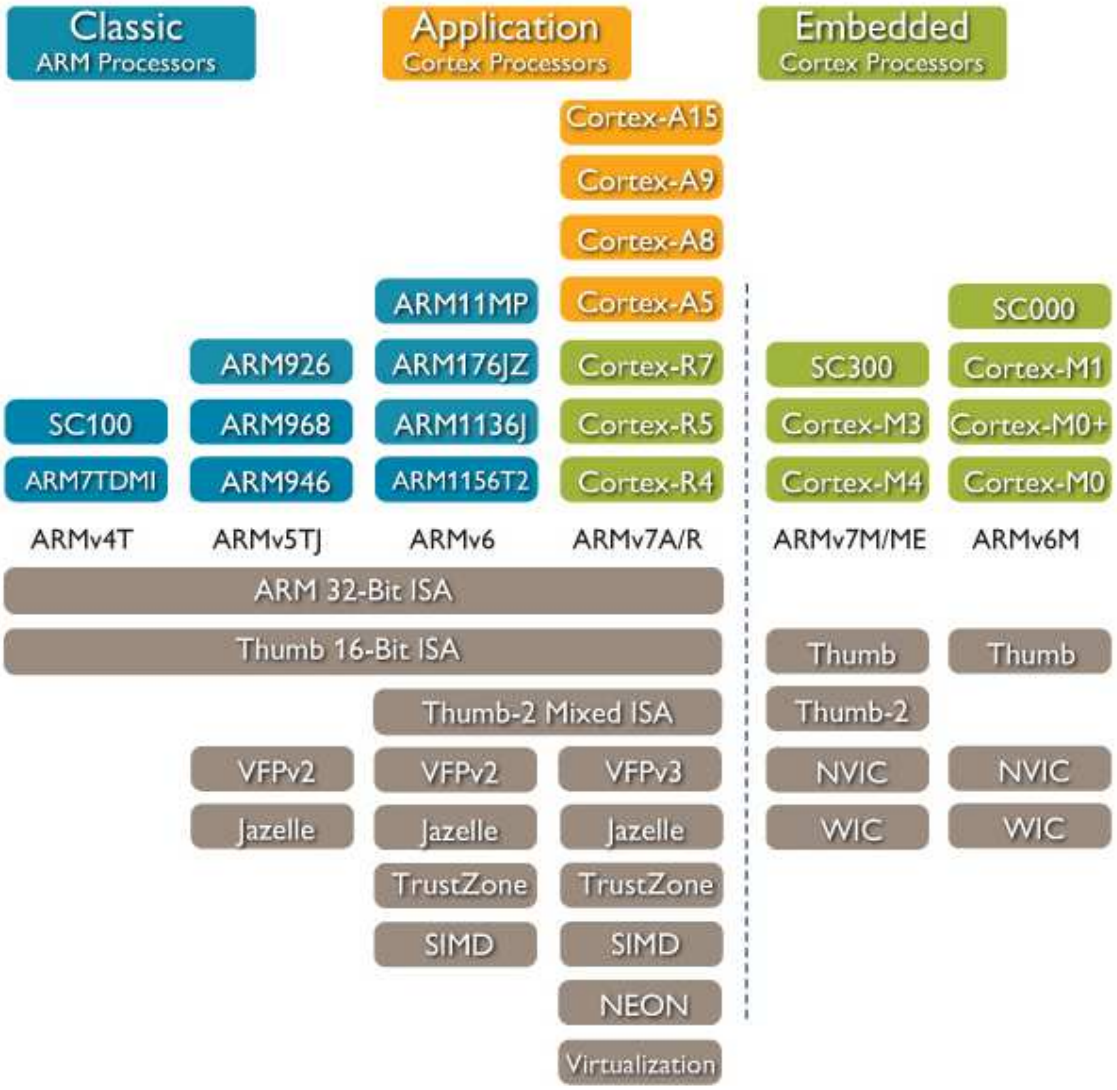
Versões da CPU ARM

- V7: geração mais recente da arquitetura ARM e totalmente baseada na tecnologia Thumb-2. Existem 3 variantes da versão:
  - A – Para alto desempenho. Para sistemas operacionais sofisticados, ou baseados em memória virtual e aplicações de usuário.
  - R – Para aplicações de tempo real.
  - M – Para pequenas aplicações de baixo custo. Otimizado para aplicações de baixo custo

Características

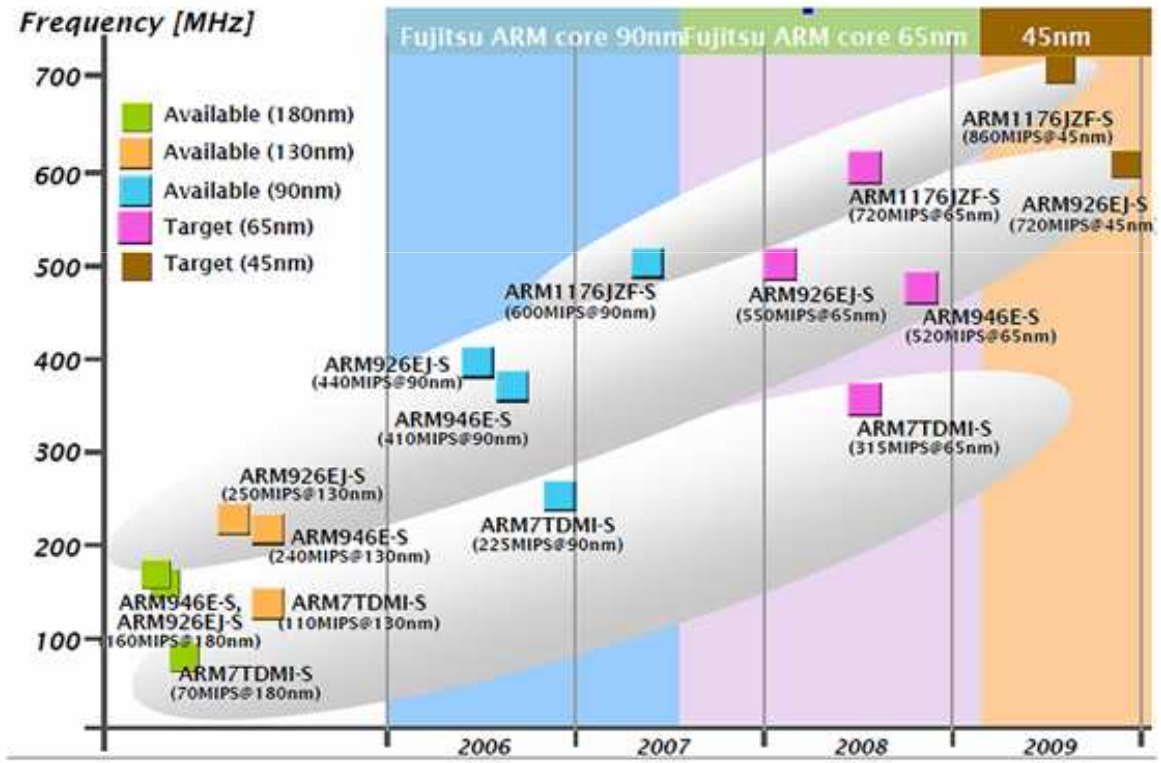
Séries ARM

Versões ARM



Características

- Modernas tecnologias



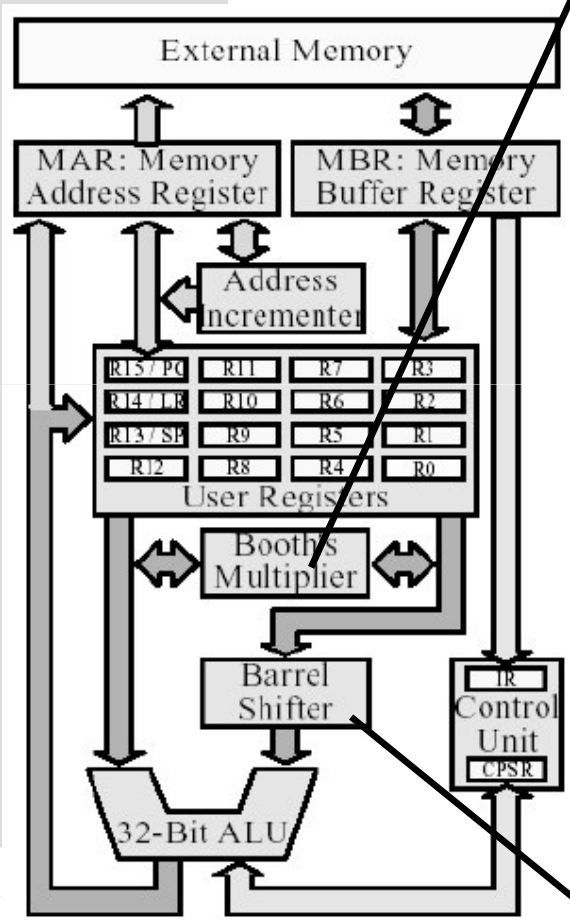
Famílias da ARM

- As diferentes versões das CPUs ARM, resultaram na criação de diversas famílias ARM, em que uma é evolução da família anterior.
- Somente das versões 4 e superiores do ARM continuam em uso.

CPU	Description	ISA	Process	Voltage	Area mm2	Power mW	Clock / MHz	Mips / MHz
ARM7TDMI	Core	V4T	0.18u	1.8V	0.53	<0.25	60-110	0.9
ARM7TDMI-S	Synthesizable core	V4T	0.18u	1.8V	<0.8	<0.4	>50	0.9
ARM9TDMI	Core	V4T	0.18u	1.8V	1.1	0.3	167-220	1.1
ARM920T	Macrocell 16+16kB cache	V4T	0.18u	1.8V	11.8	0.9	140-200	1.05
ARM940T	Macrocell 8+8kB cache	V4T	0.18u	1.8V	4.2	0.85	140-170	1.05
ARM9E-S	Synthesizable core	V5TE	0.18u	1.8V	?	~1	133-200	1.1
ARM1020E	Macrocell 32+32kB cache	V5TE	0.15u	1.8V	~10	~0.85	200-400	1.25

ARM CPU

Booth Algorithm
Algoritmo de Multiplicação que multiplica dois números binários com sinal em notação complemento de 2.



- Processador RISC de 32 bits
  - Instruções de 32 bits
- MAR: aponta a posição de memória (endereço) onde uma informação será lida ou escrita.
- MBR: contém a informação lida ou escrita na posição de memória apontada por MAR.
- 16 registradores de 32 bits (37 registradores internos)
- Pipeline (ARM7: 3 estágios)
- Cache
- Tipos de dados de 8, 16 e 32 bits.
- 7 modos de operação
  - USR,Supervisor,FIQ, IRQ, ABT, SYS, UND
- Estrutura Simples
  - Baixo consumo / Bom desempenho

Barrel Shifter
Circuito Digital que pode dar um shift em uma palavra de N bits em um único ciclo de clock.

Modos de Operação

- O processador ARM possui 7 modos de operação
  - USR (User Mode): Execução normal de programas
    - Programa em execução não pode alterar o modo de operação nem habil/desabil interrup. Apenas por reset/SWI

Para processamento de Interrupções

  - FIQ (Fast Interrupt Mode): Tratamento de interrupções rápidas
  - IRQ (Interrupt Mode): Usado para tratamento de interrupções comuns

Para utilização de tarefas e S.O.

  - Supervisor: automaticamente ativado após reset ou interrupção de software SWI.
  - SYS (System Mode): Executa rotinas privilegiadas do S.O.

Processamento de outras exceções

  - ABT (Abort Mode): Tenta tratar acesso a posições de memória (endereços) não permitidos. Dois tipos: (1) busca de instrução em endereço não permitidos (2) acesso a dado em endereço não permitido.
  - UND (Undefined Mode): invocado quando instrução não definida atinge o estágio de execução (final do pipeline).

Estados da CPU

- As CPUs ARM7 implementam a versão 4T da arquitetura ARM e podem operar com dois conjuntos de instruções:
  - **Estado ARM** (conjunto de 32 bits)
    - 34 instruções disponíveis (cada uma com largura de 32 bits)
    - Características:
      - Acesso ao conjunto completo de 16 registradores da CPU (uso geral)
      - Execução condicional de todas as instruções
      - Suporte a instruções para co-processadores

Estados da CPU

- **Estado Thumb** (conjunto de 16 bits)
  - Disponíveis 30 instruções (cada uma com largura de 16 bits)
  - Características
    - Trata-se de um subconjunto de instruções da ARM comprimidas de 32 para 16 bits
    - Estas instruções são decomprimidas em tempo real e convertidas nas instruções ARM equivalentes, antes de serem executadas.
    - Em virtude da largura reduzida:
      - » Acesso direto a apenas 8 registradores (uso geral)
      - » Inexistência de instruções condicionais (a não ser os desvios condicionais)
      - » Inexistência de instruções para co-processadores
      - » Impossibilidade de alterar o modo de processamento.
      - » Não podem se valer de instruções encontradas no modo ARM como: deslocamento de um dado, MAC, etc.

Estados da CPU

- **Estado Thumb** (conjunto de 16 bits)
  - Vantagens
    - Aumento da densidade de código (comparado ao modo ARM).
    - Uma sequência de código Thumb tende a ocupar cerca de 30% menos memória que a mesma sequência codificada com instruções ARM.

Estados da CPU

- Obs: O código no modo ARM tende a ser até **40% mais rápido**, já que são necessárias mais instruções Thumb para executar o mesmo trabalho.
- Obs: De uma maneira geral, utiliza-se o **código ARM** nas rotinas em que a **temporização é crítica**. No **resto do programa** normalmente usa-se **código Thumb**, por conta da economia de memória.
- Obs: Para provocar mudança de estado ARM <-> Thumb, o programador deve utilizar a **instrução BX** (desvio com mudança de estado), disponível nos dois conjuntos de instruções.
- No caso da Linguagem C (Compilador IAR), é possível usar os identificadores **_arm** e **_thumb** para forçar o compilador a gerar código ARM ou Thumb para a função específica.
  - Neste caso o compilador providencia o uso da instrução BX para realizar a chamada com a alteração do estado da CPU

Registradores da CPU

- A arquitetura ARM inclui um total de 37 registradores da CPU, todos com largura de 32 bits.
- A quantidade de registradores disponíveis depende do estado da CPU.
  - Estado ARM: 17 registradores (16 de uso geral e um de estado de programa)
  - Estado Thumb: 12 registradores (oito de uso geral e quatro de uso interno da CPU)

Registradores da CPU

Modo ARM	Modo *Thumb*	Função
Registrador	Registrador	
R0	R0	Uso geral
R1	R1	
R2	R2	
R3	R3	
R4	R4	
R5	R5	
R6	R6	
R7	R7	
R8		
R9		
R10		
R11		
R12		
R13	SP	Apontador da pilha (SP)
R14	LR	Endereço de retorno (LR)
R15	PC	Contador de programa (PC)
CPSR	CPSR	Estado da CPU

Registradores no estado ARM

- No estado ARM, os registradores R0 e R12 são destinados ao uso geral e podem ser utilizados livremente pelo programador (ou compilador).
- O R13 é normalmente usado como apontador de pilha de memória (SP). No estado ARM o controle da pilha fica inteiramente a cargo do sofware, que pode selecionar a sua posição e direção de empilhamento.
- O R15 é utilizado como contador de programa (PC), e aponta o endereço de memória e que a próxima instrução será buscada.
  - Observe que a memória é organizada em bytes, portanto o PC é incrementado de 4 em 4 bytes a cada nova instrução ARM executada.
- Devido ao Pipeline (busca, decodificação,execução), o PC está sempre apontando 8 bytes adiante a instrução atualmente em execução. (Em desvios o pipeline é esvaziado (flushed) e em seguida preenchido novamente -> 3 ciclos).

Registradores no estado ARM

- O **CPSR** (Current Program Status Register – Registrador de Estado Atual do Programa) contém os bits de estado do processador: flags de resultados da ULA, bits de controle de interrupção, bits de controle do modo de operação do Proc.
- Registrador **SPSR** (Saved Program Status Register – Registrador de Salvamento do Estado do Programa) armazena o temporariamente o conteúdo do CPSR durante o processamento de uma execução.
  - Na entrada da rotina de tratamento da exceção, a CPU copia automaticamente o CPSR para o SPSR.
  - Na execução de uma rotina especial de retorno do programa faz com que o conteúdo salvo no SPSR seja copiado pra o CPSR.

Registradores no estado ARM

Modo de processamento							Função
Usuário (usr)	*System* (sys)	Supervisor (svc)	*Abort* (abt)	*Undefined* (und)	IRQ (irq)	FIQ (fiq)	
R0	R0	R0	R0	R0	R0	R0	Uso geral
R1	R1	R1	R1	R1	R1	R1	
R2	R2	R2	R2	R2	R2	R2	
R3	R3	R3	R3	R3	R3	R3	
R4	R4	R4	R4	R4	R4	R4	
R5	R5	R5	R5	R5	R5	R5	
R6	R6	R6	R6	R6	R6	R6	
R7	R7	R7	R7	R7	R7	R7	
R8	R8	R8	R8	R8	R8	R8_fiq	
R9	R9	R9	R9	R9	R9	R9_fiq	
R10	R10	R10	R10	R10	R10	R10_fiq	
R11	R11	R11	R11	R11	R11	R11_fiq	
R12	R12	R12	R12	R12	R12	R12_fiq	
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq	Apontador da pilha (SP)
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq	Endereço de retorno (LR)
R15	R15	R15	R15	R15	R15	R15	Contador de programa (PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	Estado do programa
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq	Estado salvo do programa

Registradores no estado Thumb

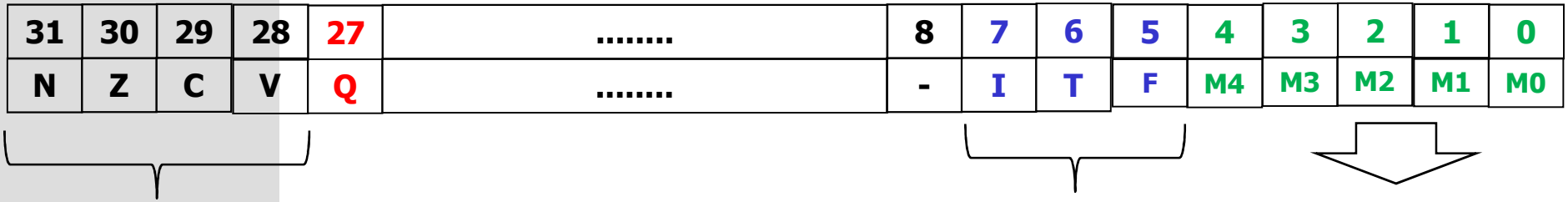
- No estado ARM, apenas 8 registradores (R0 a R7) são diretamente acessíveis à maioria das instruções.
- O R13 no modo ARM opera sempre como apontador da pilha (SP) no estado Thumb e ela é implementada como uma estrutura LIFO (Pilha).
- O registrador R14 é chamado de LR no estado Thumb e também não pode ser acessado pela maioria das funções.
- O registrador R15 no modo ARM é chamado de PC no estado Thumb e também não pode ser acessado pela maioria das instruções.
- Como as instruções Thumb possuem largura de 16 bits, o PC é incrementado de 2 em 2 bytes a cada nova instrução executada.
  - Devido ao pipeline (busca, descompressão, execução) o PC aponta sempre apontando 4 bytes adiante a instrução atualmente em execução.

Registradores no estado Thumb

Modo de processamento							Função
Usuário (usr)	*System* (sys)	Supervisor (svc)	*Abort* (abt)	*Undefined* (und)	IRQ (irq)	FIQ (fiq)	
R0	R0	R0	R0	R0	R0	R0	Uso geral
R1	R1	R1	R1	R1	R1	R1	
R2	R2	R2	R2	R2	R2	R2	
R3	R3	R3	R3	R3	R3	R3	
R4	R4	R4	R4	R4	R4	R4	
R5	R5	R5	R5	R5	R5	R5	
R6	R6	R6	R6	R6	R6	R6	
R7	R7	R7	R7	R7	R7	R7	
SP	SP	SP_svc	SP_abt	SP_und	SP_irq	SP_fiq	Apontador da pilha
LR	LR	LR_svc	LR_abt	LR_und	LR_irq	LR_fiq	Endereço de retorno
PC	PC	PC	PC	PC	PC	PC	Contador de programa
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	Estado do programa
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq	Estado salvo do programa

Registrador CPSR

- O registrador CPSR (Registrador de Estado Atual da CPU) contém os principais bits e flags de controle da CPU



N=indicador de negativo
Z =indicador de zero
C =carry
V =overflow (resultado maior que limite)

Q =Indicação de Saturação

Indicação de saturação para operações com instruções DSP.
Saturação no resultado de uma adição usando QADD ou QSUB.

I=Controle de Interrupção IRQ
F=Controle de Interrupção FIQ
T=Controle do estado da CPU

0 = estado ARM
1 = estado Thumb
Obs: No estado Usuário = Nunca alterar T
Deve-se usar o registrador BX

Modos Process CPU

Bits de Seleção do Modo de Processamento da CPU

M4	M3	M2	M1	M0	MODO
1	0	0	0	0	User
1	1	1	1	1	System
1	0	0	0	1	FIQ
1	0	0	1	0	IRQ
1	0	0	1	1	Supervisor
1	0	1	1	1	Abort
1	1	0	1	1	Undefined

Modos de Processamento

- Modo Usuário (usr)
  - O modo de usuário é não-privileiado e utilizado para operações cotidianas do programa.
  - O programa de execução não pode alterar o modo de operação nem habilitar/desabilitar interrupções
  - As únicas formas de saída desse modo não o reset ou a execução de uma interrupção por software (SWI).
  - Em ambos os casos, o novo modo de operação para a ser o supervisor.

Modos de Processamento

- Modo System (sys)
  - Esse modo privilegiado é normalmente empregado para execução de sistemas operacionais.
  - Modo privilegiado de usuário para sistema operacional
  - Permite acesso ao mesmo conjunto de registradores do modo de usuário, mas não possui as limitações dele.
  - Como este é um modo privilegiado, é possível selecional qualquer outro modo de operações a partir dele.

Modos de Processamento

- Modo Supervisor (svc)
  - O modo supervisor é automaticamente selecionado após um reset ou uma interrupção de software (SWI).
  - Normalmente utilizado para execução de sistemas operacionais.
  - Quando a entrada nesse modo foi provocada pela execução de uma instrução SWI, o endereço de retorno (o endereço da instrução seguinte a SWI) é salvo no registrador LR(R14) e o conteúdo do CPSR é salvo em SPSR_svc.
  - Em seguida a CPU selciona o modo correto nos bits M4 a M0 do registrador CPSR (10011), seleciona o estado ARM(T=0) e desabilita as interrupções IRQ(I=1).
  - Após isso, o PC(R15) é carregado com valor 0x00000008, o que faz o cluso do programa seja desviado para esse endereço. O conteúdo do pipeline esvazia-se e ele passa a ser preenchido com as instruções do novo endereço e seguintes.
  - Após a execução da instrução, o conteúdo do pipeline é novamente e carregado com as instruções seguintes a SWI.

Modos de Processamento

- Modo Abort (abt)
  - Modo privilegiado utilizado para processamento dos chamados aborts, que consistem basicamente na tentativa de acesso a posições de memória não implementadas ou não permitidas.
  - Existem duas modalidades de abort, sendo uma por **busca de instrução em endereço não permitido** (prefetch abort) e outra por **acesso de dado** em endereço não permitido (data abort).
  - Prefetch Abort
    - Esse tipo de abort somente ocorre ao tentar executar uma instrução, biscada em uma posição de memória inválida ou não permitida.
  - Data Abort
    - Possui comportamento similar ao prefetch abort, com a diferença de que esse tipo de abort é provocado pela tentativa de escrever ou ler um adado em uma posição de memória não permitida.

Modos de Processamento

- Modo Undefined (und)
  - O undefined é um modo de processamento privilegiado invocado quando uma instrução não definica atinge a unidade de execução (o final do pipeline).
  - Quando a CPU não reconhece a instrução, a linha interna nCPI é ativada (nível 0), o caso o co-processador esteja presente e reconheça a instrução, ele sinaliza este fato à CPU (através da linha CPA). A CPU então aguarda que o co-processador termine de executar a instrução, o que é sinalizado pela linha interna CPB. Nesse caso nenhuma exceção é gerada.

Modos de Processamento

- Modo IRQ (irq)
  - Modo privilegiado para tratamento de interrupções nos ARM.
  - Ele é invocado quando a linha de interrupção IRQ da CPU é colocada em nível lógico “0” ao mesmo tempo em que o bit I do CPSR está em nível “0”.
  - Reconhecida a interrupção, a CPU completa a execução da instrução corrente e realiza as ações:
    - O endereço da próxima instrução mais 4 é armazenado no R14_irq
    - O conteúdo do CPSR é armazenado no registrador SPSR_irq
    - O registrador CPSR é alterado. O modo IRQ é selecionado nos bits de modo (10010), assim como o estado ARM (T=0). A interrupção IRQ é desabilitada (I=1).
    - Em seguida o PC é carregado com o endereço 0x00000018 e o programa desviado para lá.

Modos de Processamento

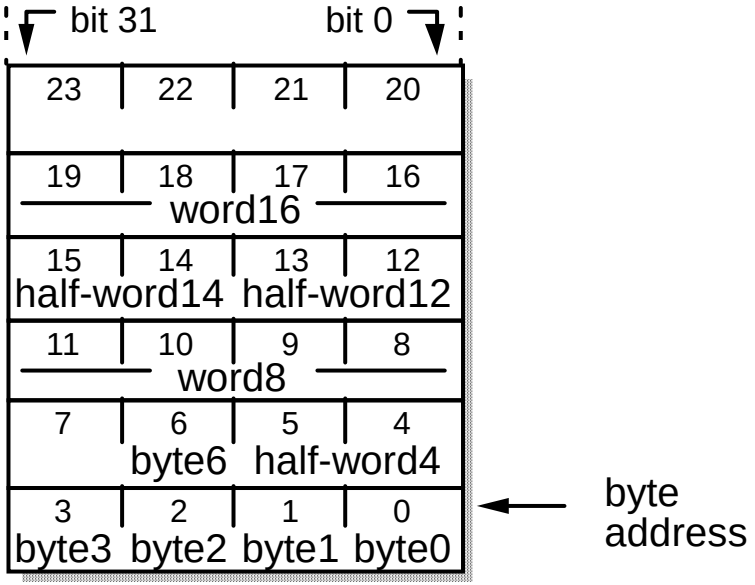
- Modo FIQ (fiq)
  - Outro modo de interrupção disponível nos ARMs.
  - Diferenças entre ele e o IRQ são:
    - **Maior prioridade**: o modo FIQ possui maior prioridade que o IRQ.
    - **Conjunto de registradores especiais**: o modo FIQ dispõe de um conjunto de registradores R8 a R12 diferenciado do modo IRQ e dos demais modos de operação da CPU. Isso permite que o tratamento da interrupção responda muito **mais rápido** ao evento da interrupção, já que esses registradores podem ser empregados localmente sem a necessidade de preservação do seu conteúdo.

Organização de Memória do ARM

- Apesar de se tratar de uma CPU de 32 bits, o ARM é capaz de manipular dados de 8 e 16 bits.
- Além disso o conjunto de instruções Thumb possui tamanho de 16 bits.
- O contador de programa (PC) desses chips aponta sempre para *words* de memória (essa é a razão de ser incrementado de 2 em 2 a cada instrução Thumb e de 4 em 4 para ARM).
- Também suportam organização de dados no formato **little endian** e **big endian**.

Organização de Memória do ARM

- Array linear de bytes numerados de 0 a $2^{32} - 1$
- A CPU normalmente requer alinhamento da memória conforme o tipo de dado.
- Tipos de dados
  - bytes (8 bits)
  - half-words (16 bits) – sempre alinhadas no limite de 2-bytes (iniciam em endereço par)
  - words (32 bits) – sempre alinhadas no limite de 4-bytes (iniciam em endereço múltiplo de 4)



Barrel Shifter