HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG MÔN

KỸ THUẬT VI XỬ LÝ

Giảng viên: TS. Nguyễn Trung Hiếu

Điện thoại/E-mail: 0916566268 / hieunt@ptit.edu.vn

Học kỳ: Kỳ 1/2020

KỸ THUẬT VI XỬ LÝ

NỘI DUNG

- Chương 1 – Tổng quan về hệ vi xử lý
- Chương 2 – Bộ vi xử lý ARM
- Chương 3 – Lập trình hợp ngữ cho vi xử lý ARM
- Chương 4 – Vi điều khiển 8051
- Chương 5 – Bộ đếm/định thời và UART trong 8051
- Chương 6 – Lập trình ngắt trong 8051

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

NỘI DUNG

1. Giới thiệu về ARM Ltd
2. Vi xử lý ARM
    1. Giới thiệu
    2. Mô hình lập trình
    3. Vi xử lý ARM7TDMI
    4. Dòng chảy tác vụ

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

- Thành lập 11/1990
  - Được tách ra từ Acorn và trở thành Advanced RISC Machines Ltd
  - VXL đầu tiền được tạo ra cho Newton PDA
- VXL ARM sử dụng tập lệnh RISC. Vì vậy ARM cũng có nghĩa là Advanced RISC Machines.
- Hiện nay số lượng nhân sự của ARM lên tới hơn 1500 người. ARM có nhiều chi nhánh tại nhiều quốc gia.

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

- ARM hỗ trợ nhiều dòng sản phẩm và hệ điều hành bao gồm Symbian, Palm, Windows, Linux

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

- Mô hình sản xuất kinh doanh của ARM gồm hơn 40 đối tác chính để sản xuất các linh kiện bán dẫn cho ARM.
- Ngoài ra, ARM có rất nhiều đối tác khác để phát triển hệ điều hành, các ứng dụng trên bộ vi xử lý ARM.

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

- Công ty ARM không chỉ thiết lập các chuẩn trong nội bộ ARM mà còn cho toàn bộ nền công nghiệp sản xuất VXL.
- Các chi nhánh của ARM trải rộng trên toàn cầu.

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

NỘI DUNG

1. Giới thiệu về ARM Ltd
2. Vi xử lý ARM
    1. Giới thiệu
    2. Mô hình lập trình
    3. Tập lệnh

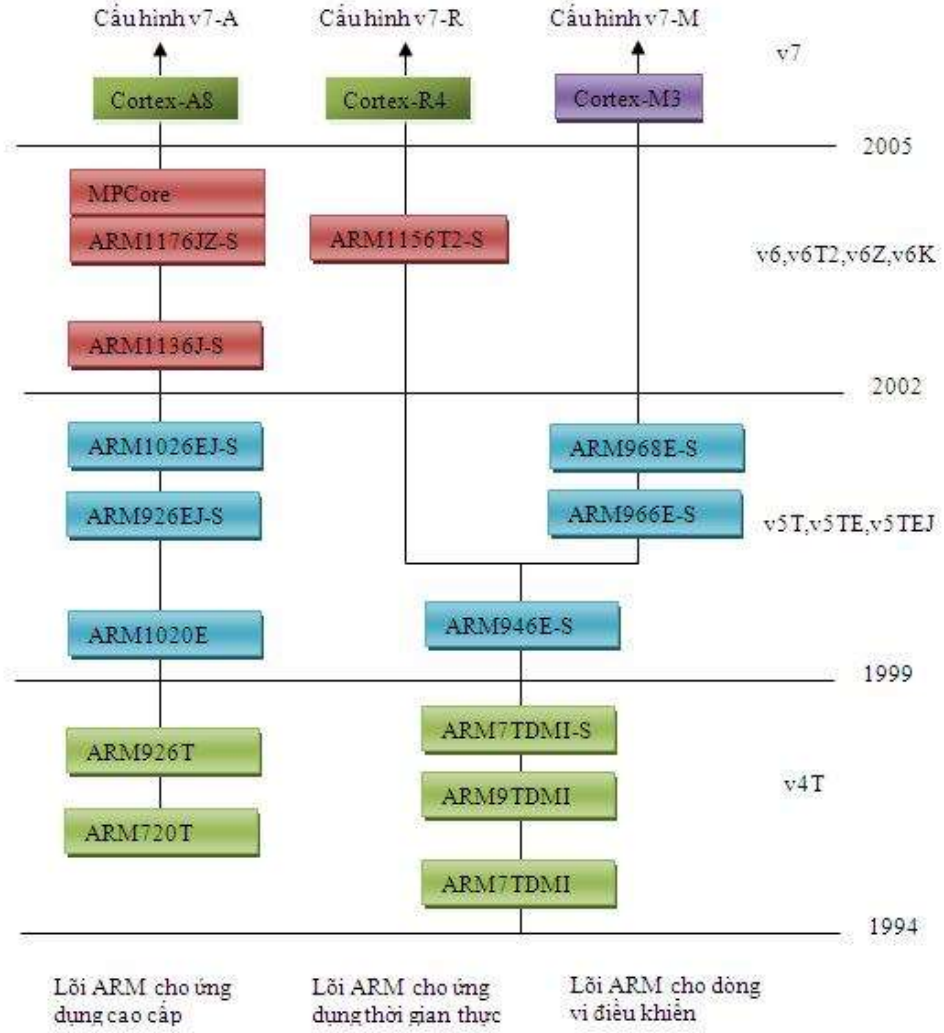
CHƯƠNG 2-BỘ VI XỬ LÝ ARM

LỊCH SỬ KIẾN TRÚC ARM

- Việc thiết kế ARM được bắt đầu từ 1983 bởi công ty Acorn
- Nhóm hoàn thành việc phát triển ARM1 vào năm 1985 và ARM2 vào năm 1986. ARM2 có bus dữ liệu 32 bit, bus địa chỉ 26 bit, 16 thanh ghi 32 bit.
- Cuối thập niên 80, Acorn nâng cấp nhóm phát triển ARM thành công ty Advanced RISC Machine – ARM và cho ra đời ARM6.
- Đầu thập niên 90, ARM phát triển VXL ARM7TDMI (T: Thumb, D: Debux, M: Long Multiply Support, I: EmbeddedICE macrocell)

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

- Lịch sử kiến trúc ARM



ARM® Cortex®-A Current Portfolio

High Efficiency

ARMv8-A
highest efficiency 64/32-bit CPU

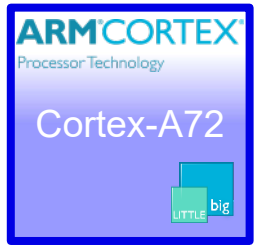
ARMv7-A
smallest and lowest power CPU

ARMv7-A
highest efficiency 32-bit CPU

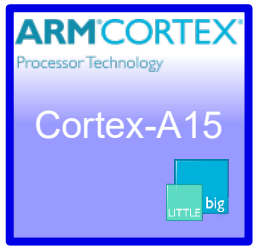
ARMv7-A 32-bit CPU
Shipping since 2009

High Performance

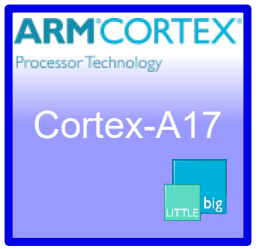
ARMv8-A
64/32-bit proven high-end CPU



ARMv8-A
highest 64/32-bit performance CPU



ARMv7-A
32-bit performance with enterprise class feature set



ARMv7-A
high performance 32-bit only CPU for mobile and consumer

ARM® Cortex®-A Current Portfolio

- Dòng vxl ứng dụng ARM (Application Processor) được sử dụng để chạy các hệ điều hành phức tạp bao gồm Linux, Android, Chrome OS, Tizen, Microsoft Windows (CE/Embedded)...
- Các thiết bị sử dụng lớp vxl này bao gồm Smartphones, Tablets / eReaders, Digital Television , Set-top Boxes & Satellite Receivers, Server, Wearables,...
- Thế hệ vxl Cortex-A5, Cortex-A7, Cortex-A9, Cortex-A15 và Cortex-A17 được phát triển dựa trên kiến trúc của thế hệ vxl ARMv7-A. Đây là các thế hệ vxl có tập lệnh 32 bit.
- Các thế hệ vxl mới bao gồm Cortex-A72, Cortex-A53 và Cortex-A57 được phát triển từ thế hệ vxl Cortex-A với tập lệnh 64 bit. Các thế hệ này cũng chạy được tập lệnh 32 bit.

ARM® Cortex®-R and Cortex®-M Processor Portfolio

M Cortex-M

Mainstream
Control & DSP

Maximum Performance
Control & DSP

Low power with
highest cost
efficiency

Highest energy
efficiency

Performance
efficiency

R Cortex-R

High performance
4G modem and storage

Real-time standard

Functional safety
package

ARM® Cortex®-R and Cortex®-M Processor Portfolio

- Dòng vxl Cortex-M được sử dụng cho các các ứng dụng máy tính nhúng công suất thấp. Thông thường, các thiết bị sử dụng Cortex-M được sử dụng như những “hộp đen” với các ứng dụng định sẵn. Các thiết bị này hạn chế trong việc mở rộng chức năng và không có màn hình.
- Thiết bị sử dụng Cortex-M bao gồm các hệ điều khiển tự động, các thiết bị đo thông minh, cảm biến,...
- Dòng vxl Cortex-R được dùng để xử lý các ứng dụng thời gian thực. Các ứng dụng cho lớp vxl này bao gồm: Các hệ thống điều khiển tự động, mạng cảm biến không dây và có dây, hạ tầng các trạm gốc vô tuyến, thiết bị mạng,...

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

NỘI DUNG

1. Giới thiệu về ARM Ltd
2. Vi xử lý ARM
    1. Giới thiệu
    2. Mô hình lập trình
    3. Tập lệnh

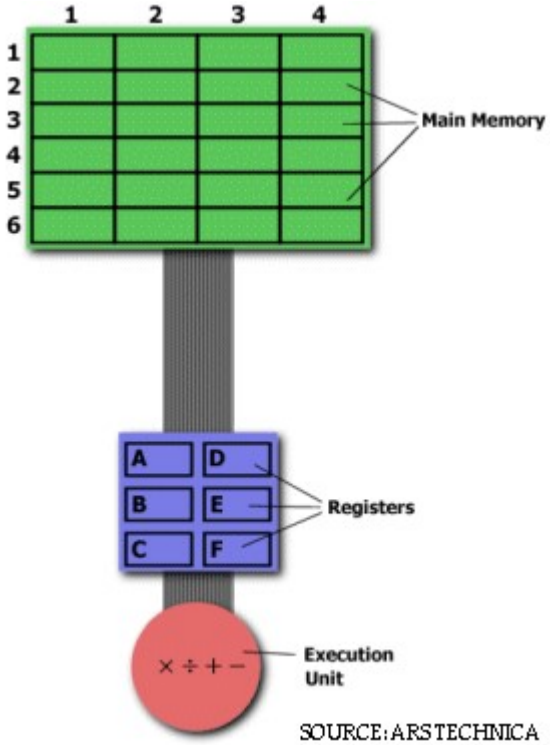
CHƯƠNG 2-BỘ VI XỬ LÝ ARM

MỘT SỐ KHÁI NIỆM

- CISC (Complex Instruction Set Computer) vs. RISC (Reduced Instruction Set Computer)
- CISC: MULT 2:3, 5:2
- RISC: MOV A, 2:3

  MOV B, 5:2

  MUL A,B
- Ưu điểm của RISC:
  - Kích thước bán dẫn nhỏ
  - Thời gian phát triển sản phẩm ngắn hơn
  - Cấu hình mạnh hơn



CHƯƠNG 2-BỘ VI XỬ LÝ ARM

MỘT SỐ KHÁI NIỆM

- Kiến trúc Load/Store:
  - Các toán hạng của các lệnh (cộng, trừ, …) đều được lưu và xử lý trên thanh ghi.
  - Chỉ có lệnh copy giá trị từ bộ nhớ vào thanh ghi (load) hoặc chép lại giá trị từ thanh ghi vào bộ nhớ (store) mới ảnh hưởng tới bộ nhớ.
  - Bộ xử lý CISC cho phép xử lý dữ liệu trực tiếp trên bộ nhớ.

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

KÍCH THƯỚC DỮ LIỆU VÀ TẬP LỆNH

- Vi xử lý ARM sử dụng kiến trúc RISC:
  - Rất nhiều lệnh được thực hiện chỉ trong 1 chu kỳ máy.
- ARM sử dụng kiến trúc load/store 32 bit
- Kích thước dữ liệu trong ARM:
  - Halfword: 16 bit
  - Word: 32 bit
- Hầu hết các phiên bản ARM sử dụng 2 loại tập lệnh:
  - Tập lệnh ARM 32 bit
  - Tập lệnh Thumb 16 bit
- Các phiên bản mới của ARM sử dụng thêm Thumb-2 (Phối hợp cả hai loại trên)

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

CÁC CHẾ ĐỘ CỦA VI XỬ LÝ

- Vi xử lý ARM cung cấp 7 chế độ hoạt động:
  - Mỗi chế độ có ngăn xếp và các thanh ghi riêng
  - Một số phép toán chỉ có thể thực hiện trong chế độ đặc quyền (privileged mode)

	Mode	Description	
Exception modes	Supervisor (SVC)	Entered on reset and when a Software Interrupt instruction (SWI) is executed	Privileged modes
	FIQ	Entered when a high priority (fast) interrupt is raised	
	IRQ	Entered when a low priority (normal) interrupt is raised	
	Abort	Used to handle memory access violations	
	Undef	Used to handle undefined instructions	
	System	Privileged mode using the same registers as User mode	
	User	Mode under which most Applications / OS tasks run	Unprivileged mode

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

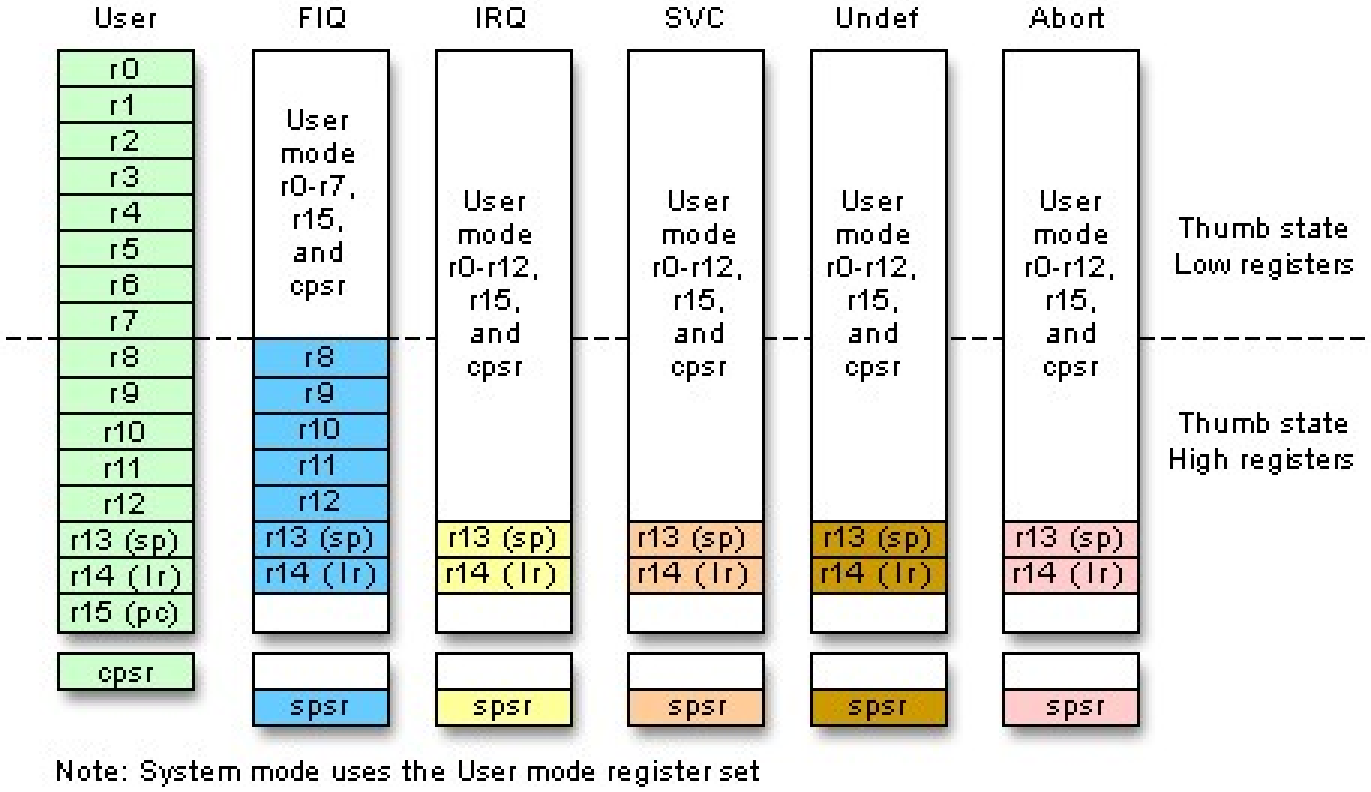
CÁC CHẾ ĐỘ CỦA VI XỬ LÝ

- Vi xử lý ARM cung cấp 7 chế độ hoạt động:
  - Bộ VXL hoạt động ở chế độ Abort khi bộ VXL không thể truy cập bộ nhớ.
  - Bộ VXL hoạt động ở chế độ interrupt request (IRQ) và fast interrupt request (FIQ) tương ứng với hai mức ngắt của chip ARM.
  - Bộ VXL hoạt động ở chế độ Supervisor sau khi hệ thống khởi động (reset) và khi nhân của hệ điều hành hoạt động.
  - Bộ VXL hoạt động ở chế độ System khi hệ thống có thể truy cập và đọc, ghi toàn bộ thanh ghi CPRS.
  - Bộ VXL chuyển sang chế độ Undefined khi bộ VXL gặp một lệnh không xác định hoặc không được hỗ trợ.
  - Bộ VXL hoạt động ở chế độ User là để chạy các chương trình và các ứng dụng thông thường.

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

TẬP THANH GHI CỦA ARM

- Vi xử lý có 37 thanh ghi. Tất cả các thanh ghi có chiều dài 32 bit.
- Mỗi chế độ bao gồm một tập các thanh ghi riêng.



Note: System mode uses the User mode register set

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

TẬP THANH GHI CỦA ARM

- Tại một thời điểm chỉ có 15 thanh ghi đa dụng (từ R0 đến R14) và 2 thanh ghi hiển thị trạng thái được sử dụng tùy theo chế độ.
- (R0 – R7) luôn được sử dụng trong tất cả các chế độ.
- (R8 - R12) cũng được sử dụng trong hầu hết các chế độ ngoại trừ chế độ ngắt nhanh (fast interrupt – fiq).
- Các thanh ghi còn lại (R13 – R15) là 3 thanh ghi có các chức năng hoặc nhiệm vụ đặc biệt riêng:
  - Thanh ghi r13 được dùng làm con trỏ ngăn xếp stack pointer (SP)
  - Thanh ghi r14 được gọi là thanh ghi kết nối (LR) chứa địa chỉ quay lại của chương trình khi chương trình chạy một hàm con.
  - Thanh ghi r15 là bộ đếm chương trình (PC) và chứa địa chỉ của lệnh tiếp theo.

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

TẬP THANH GHI CỦA ARM

- Stack pointer

```
AREA Example1,CODE,READONLY
ENTRY
MOV R2,#0x00000007          ; R2=7
MOV R1, R2                  ; R1 = R2 = 7
PUSH {R1}                   ; SP = 0xFFFFFFFC
END
```

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

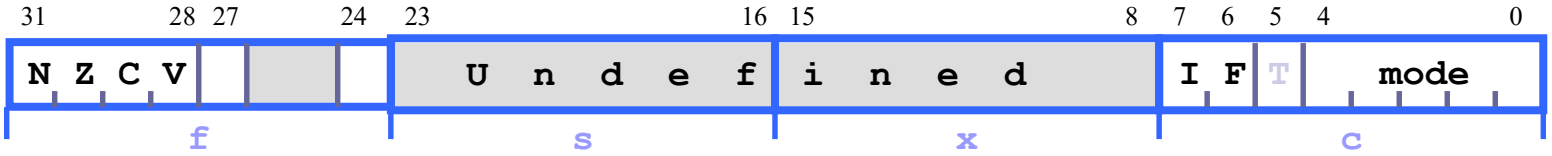
TẬP THANH GHI CỦA ARM

- Stack pointer

```
AREA Example1,CODE,READONLY
ENTRY
MOV R2,#0x00000007          ; R2=7
MOV R1, R2                  ; R1 = R2 = 7
PUSH {R1, R2}               ; SP = ?
END
```

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

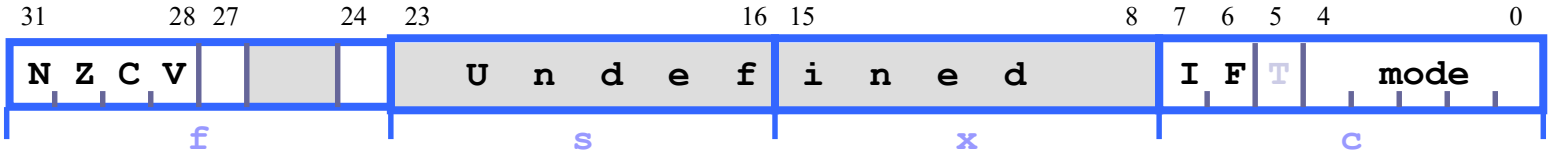
THANH GHI TRẠNG THÁI - CPSR



- Condition code flags
  - N = **N**egative result from ALU
  - Z = **Z**ero result from ALU
  - C = ALU operation **C**arried out
  - V = ALU operation o**V**erflowed
- Interrupt Disable bits.
  - I = 1: Disables the IRQ.
  - F = 1: Disables the FIQ.
- T Bit
  - Architecture xT only
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state
- Mode bits
  - Specify the processor mode

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

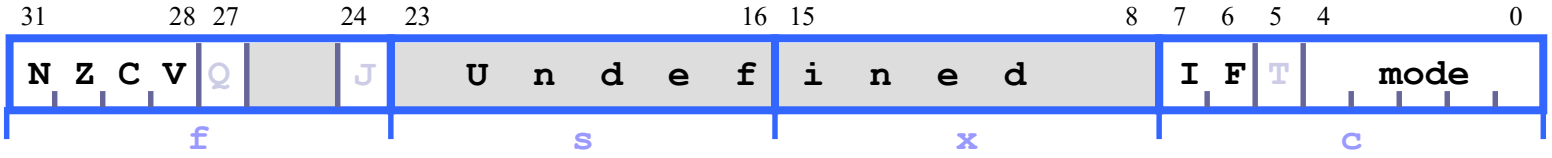
THANH GHI TRẠNG THÁI - CPSR



- Cờ nhớ C = 1 khi có nhớ sinh ra bởi phép cộng hoặc khi có mượn trong phép trừ
- Cờ tràn V = 1 khi:
  - Có nhớ từ bit 30 sang bit 31 nhưng không có nhớ ra từ bit 31.
  - Có nhớ ra từ bit 31 nhưng không có nhớ từ bit 30 sang

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

THANH GHI TRẠNG THÁI - CPSR



M[4:0]	Chế độ	Các thanh ghi được phép truy nhập
10000	User	PC, R0 – R14, CPSR
10001	FIQ	PC, R0 – R7, R8_fiq – R14_fiq, CPSR, SPSR_fiq
10010	IRQ	PC, R0 – R12, R13_irq, R14_irq, CPSR, SPSR_irq
10011	SVC	PC, R0 – R12, R13_svc, R14_svc, CPSR, SPSR_svc
10111	Abort	PC, R0 – R12, R13_abt, R14_abt, CPSR, SPSR_abt
11011	Undef	PC, R0 – R12, R13_und, R14_und, CPSR, SPSR_und
11111	System	PC, R0 – R14, CPSR

THANH GHI TRẠNG THÁI - CPSR

- Tìm giá trị của thanh ghi CPSR

```
AREA Example1,CODE,READONLY
ENTRY
MOV R2,#0x00000005          ; R2=5
MOV R1,#0x00000005          ; R1=5
ADDS R1, R2
END
```

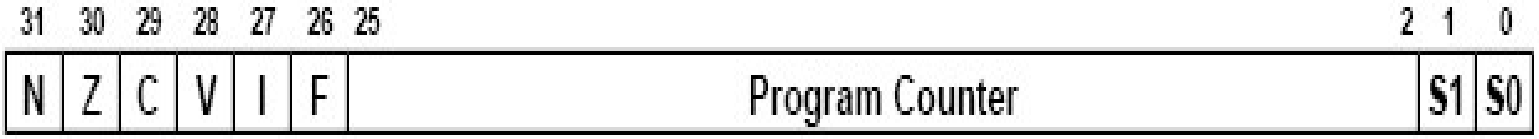
THANH GHI TRẠNG THÁI - CPSR

- Tìm giá trị của thanh ghi CPSR

```
AREA Example1,CODE,READONLY
ENTRY
MOV R2,#-0x00000005          ; R2=-5
MOV R1,#0x00000005           ; R1=5
ADDS R1, R2
END
```

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

THANH GHI CON TRỎ LỆNH - PC



- When the processor is executing in ARM state:
  - All instructions are 32 bits wide
  - All instructions must be word aligned
  - Therefore the **pc** value is stored in bits [31:2] with bits [1:0] undefined (as instruction cannot be halfword or byte aligned).

- When the processor is executing in Thumb state:
  - All instructions are 16 bits wide
  - All instructions must be halfword aligned
  - Therefore the **pc** value is stored in bits [31:1] with bit [0] undefined (as instruction cannot be byte aligned).

THANH GHI CON TRỎ LỆNH - PC

```
        AREA Example1,CODE,READONLY
        ENTRY
        MOV R2,#0x00000007          ; R2=7
        MOV R1, R2                  ; R1 = R2 = 7
        BL func1                    ; Gọi func1
        MOV R3,R2
func1
        MOV R4,R2                   ; R4 = R2 = 7
        BX LR                       ; Quay trở lại chương trình chính
        END
```

Giá trị PC = ?

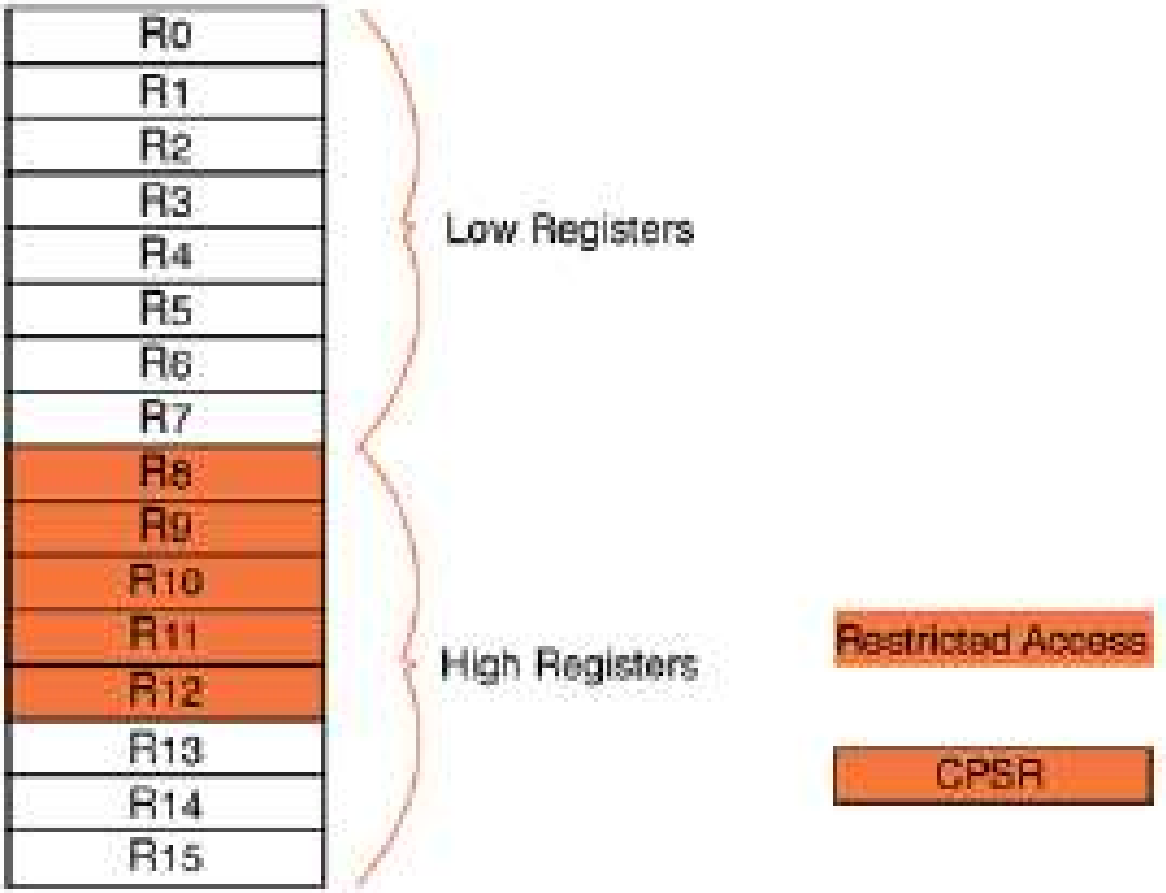
THANH GHI CON TRỎ LỆNH - PC

Tìm giá trị PC, R1, R2, R3, R4,R5 khi kết thúc chương trình

```
        AREA Example2,CODE,READONLY
        ENTRY
        MOV32 R2,#0x00000007        ; R2=7
        MOV R1, R2                  ; R1 = R2 = 7
        MOV R3, R1
        BL func1                    ; Gọi func1
        MOV R5,R2
func1
        MOV R4,R2                   ; R4 = R2 = 7
        BX LR
        END
```

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

CHẾ ĐỘ THUMB



CHƯƠNG 2-BỘ VI XỬ LÝ ARM

CHẾ ĐỘ THUMB

- The Thumb instruction set consists of 16-bit instructions that act as a compact shorthand for a subset of the 32-bit instructions of the standard ARM.
- Why have two instruction sets in the same CPU?
- The biggest register difference involves the SP register. The Thumb state has unique stack mnemonics (PUSH, POP) that don't exist in the ARM state.
- Chuyển từ ARM sang Thumb và ngược lại:

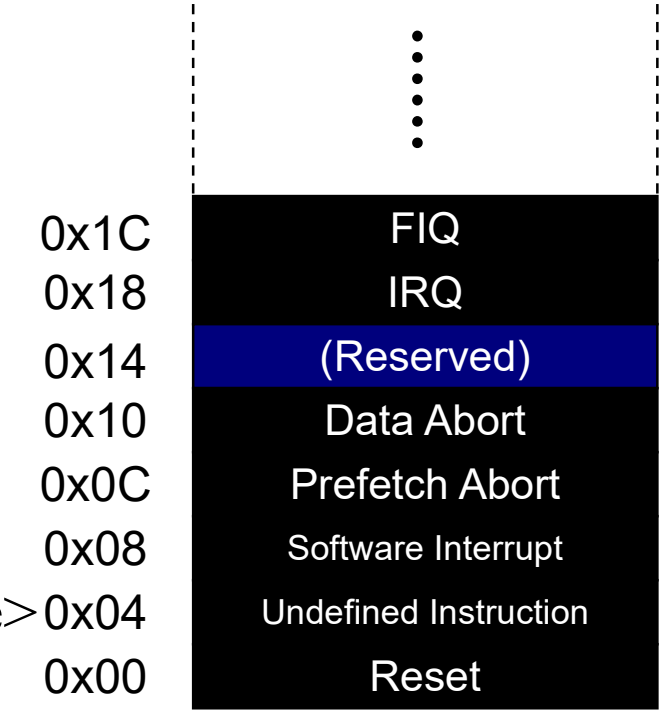
```
        .....
        LDR R0,=Thumb_mode+1;
        BX R0
Thumb_mode
        .....
        ADR R5,Arm_mode
        BX R5
Arm_mode
```

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

XỬ LÝ BIỆT LỆ

- When an exception occurs, the ARM:
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM state
    - Change to exception mode
    - Disable interrupts (if appropriate)
  - Stores the return address in LR_<mode>
  - Sets PC to vector address
- To return, exception handler needs to:
  - Restore CPSR from SPSR_<mode>
  - Restore PC from LR_<mode>

This can only be done in ARM state.



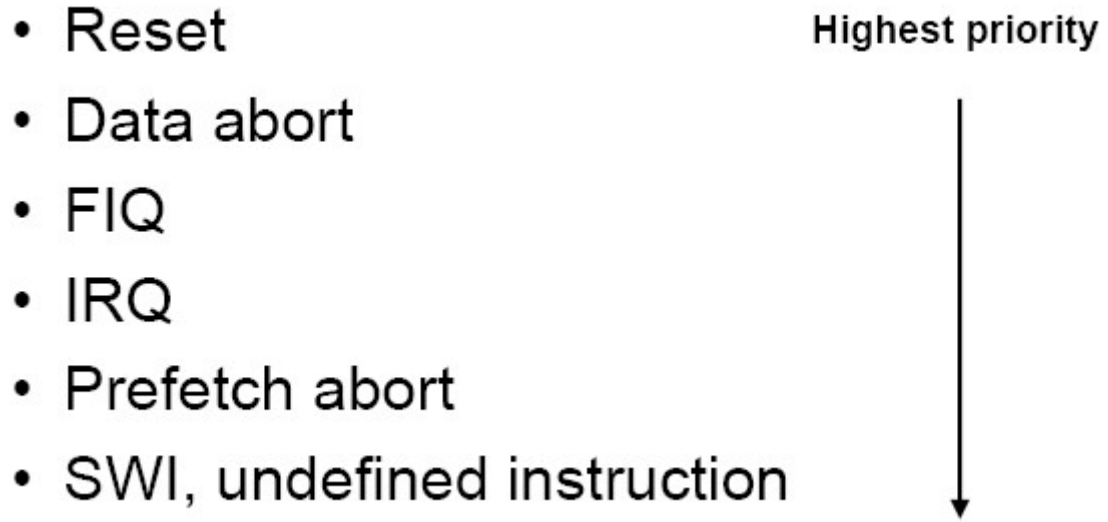
Vector Table

Vector table can be at 0xFFFF0000 on ARM720T and on ARM9/10 family devices

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

XỬ LÝ BIỆT LỆ

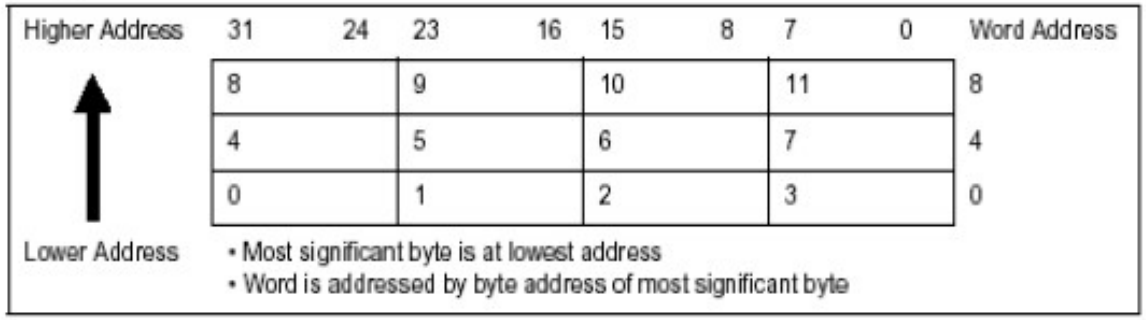
- Thứ tự ưu tiên các ngắt:
  - Prefetch abort occurs when the processor attempts to execute an instruction that has prefetched from an illegal address, that is, an address that the memory management subsystem has determined is inaccessible to the processor in its current mode.
  - Data abourt when a data transfer instruction attempts to load or store data at an illegal address.



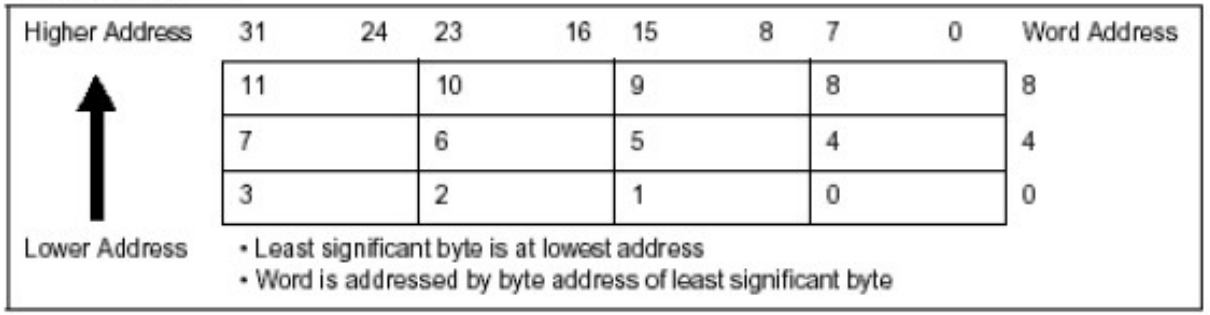
CHƯƠNG 2-BỘ VI XỬ LÝ ARM

ENDIAN CONFIGURATION

Big endian



Little endian



- Các bộ VXL ARM mặc định ở chế độ Little endian. Tuy nhiên có thể cấu hình để truy cập bộ nhớ theo chế độ Big endian.

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

ENDIAN CONFIGURATION

- Liệt kê tên các chế độ hoạt động của VXL ARM
- VXL ARM có tất cả bao nhiêu thanh ghi?
- Thanh ghi R13 có chức năng gì?
- Chế độ nào có quyền truy nhập ít thanh ghi nhất?
- Các bit nào trong thanh ghi CPSR phản ánh trạng thái của VXL?
- Liệt kê tên các biệt lệ của ARM

ENDIAN CONFIGURATION

- **Bài 1: CPSR = ?**

  MOV R2,#0xF0000000

  MOV R1,#0xF0000000

  ADDS R1,R2

- **Bài 2: CPSR = ?**

  MOV R2,#-0x80000000

  MOV R1,#0x90000000

  ADDS R1,R2

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

NỘI DUNG

1. Giới thiệu về ARM Ltd
2. Vi xử lý ARM
    1. Giới thiệu
    2. Mô hình lập trình
    3. Vi xử lý ARM7TDMI
    4. Dòng chảy tác vụ

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

VI XỬ LÝ ARM7TDMI

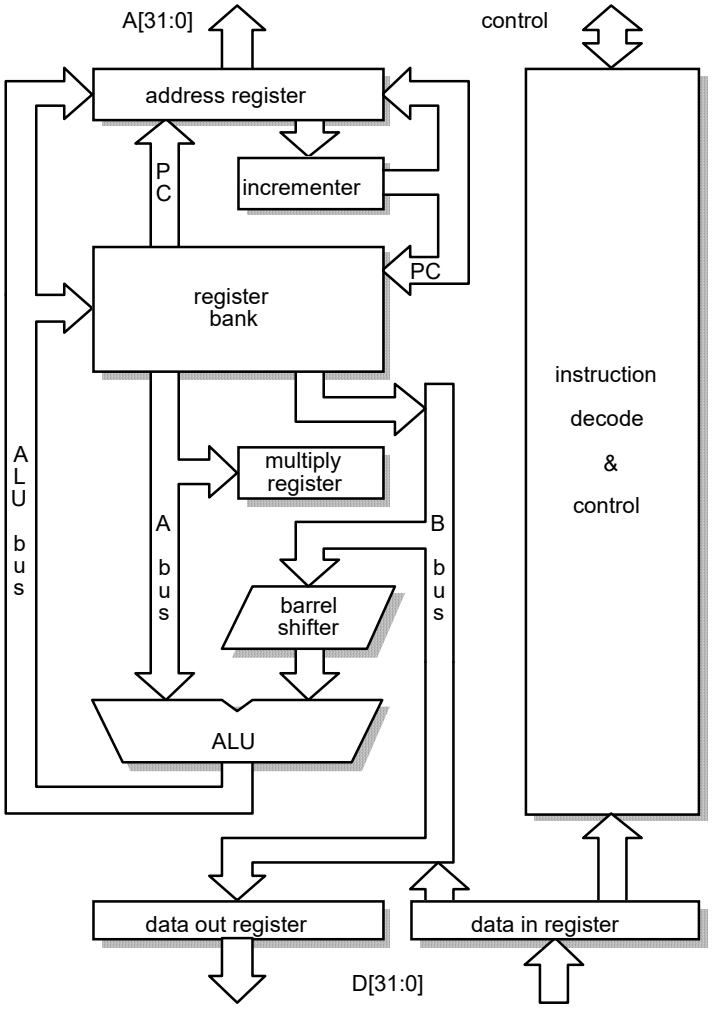
- Đặc điểm:
  - 3 tác vụ trong dòng chảy lệnh (đọc, giải mã và thực hiện)
  - Kiến trúc Von Neumann
  - CPI (Cycle per Instruction) ~ 1.9 (ARM9: CPI ~ 1.5)
  - T: Hỗ trợ tập lệnh thumb
    - Tập lệnh ARM: 32 bit
    - Tập lệnh Thumb: 16 bit
  - D: Hỗ trợ gỡ lỗi
  - M: Hỗ trợ phép nhân với kết quả 64 bit
  - I: EmbeddedICE macrocell (Hỗ trợ gỡ lỗi trên JTAG)

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

VI XỬ LÝ ARM7TDMI

- Register bank
  - 2 cổng đọc, 1 cổng ghi
  - 1 cổng đọc và 1 cổng ghi riêng cho R15
- Barrel shifter
  - Dịch hoặc quay toán hạng
- ALU
- Thanh ghi địa chỉ và bộ tăng
- Các thanh ghi dữ liệu
  - Lưu dữ liệu nhận từ bộ nhớ

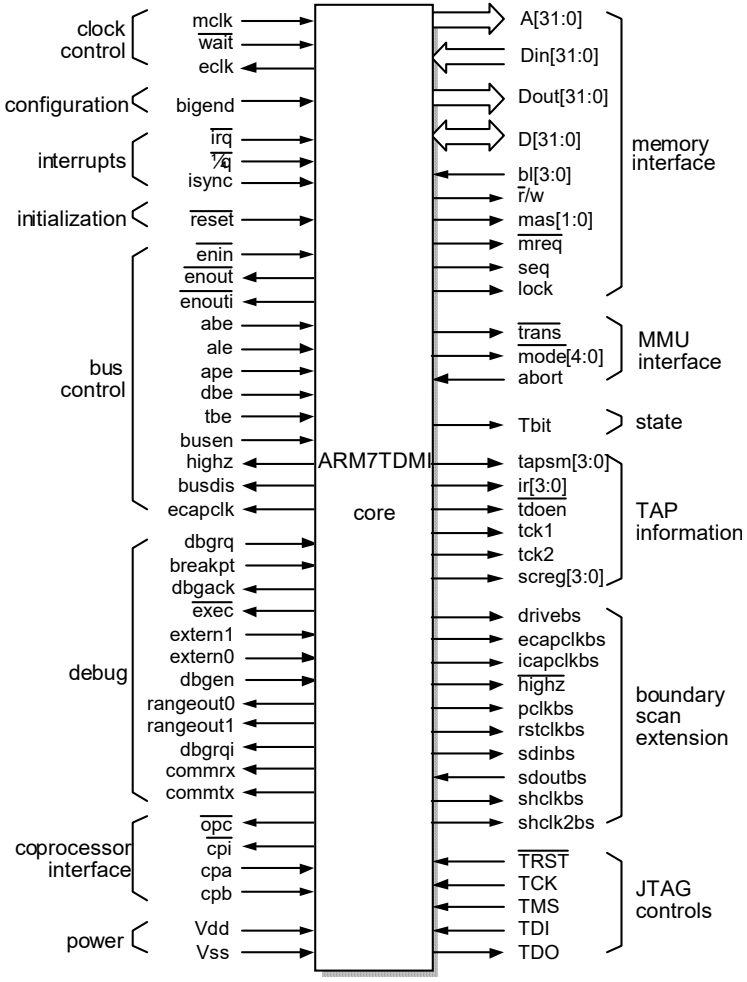
  hoặc gửi đến bộ nhớ
- Bộ giải mã lệnh và điều khiển



CHƯƠNG 2-BỘ VI XỬ LÝ ARM

VI XỬ LÝ ARM7TDMI

- Các chân tín hiệu của ARM7TDMI



CHƯƠNG 2-BỘ VI XỬ LÝ ARM

VI XỬ LÝ ARM7TDMI

- Clock control
  - All state change within the processor are controlled by *mclk*, the memory clock
  - Internal clock = *mclk* AND *wait*
- Memory interface
  - 32-bit address *A[31:0]*, bidirectional data bus *D[31:0]*, separate data out *Dout[31:0]*, data in *Din[31:0]*
  - *seq* indicates that the memory address will be sequential to that used in the previous cycle
  - Lock indicates that the processor should keep the bus to ensure the atomicity of the read and write phase of a SWAP instruction
  - \r/w, read or write
  - mas[1:0], encode memory access size – byte, half-word or word
  - bl[3:0], externally controlled enables on latches on each of the 4 bytes on the data input bus

VI XỬ LÝ ARM7TDMI

- Sequential (S cycle)
  - (nMREQ, SEQ) = (0, 1)
  - The ARM core requests a transfer to or from an address which is either the same, or one word or one-half-word greater than the preceding address.
- Non-sequential (N cycle)
  - (nMREQ, SEQ) = (0, 0)
  - The ARM core requests a transfer to or from an address which is unrelated to the address used in the preceding address.
- Idle (I cycle)
  - (nMREQ, SEQ) = (1, 0)
  - The ARM core does not require a transfer, as it performing an internal function, and no useful prefetching can be performed at the same time
- Coprocessor register transfer (C cycle)
  - (nMREQ, SEQ) = (1, 1)
  - The ARM core wished to use the data bus to communicate with a coprocessor, but does not require any action by the memory system.

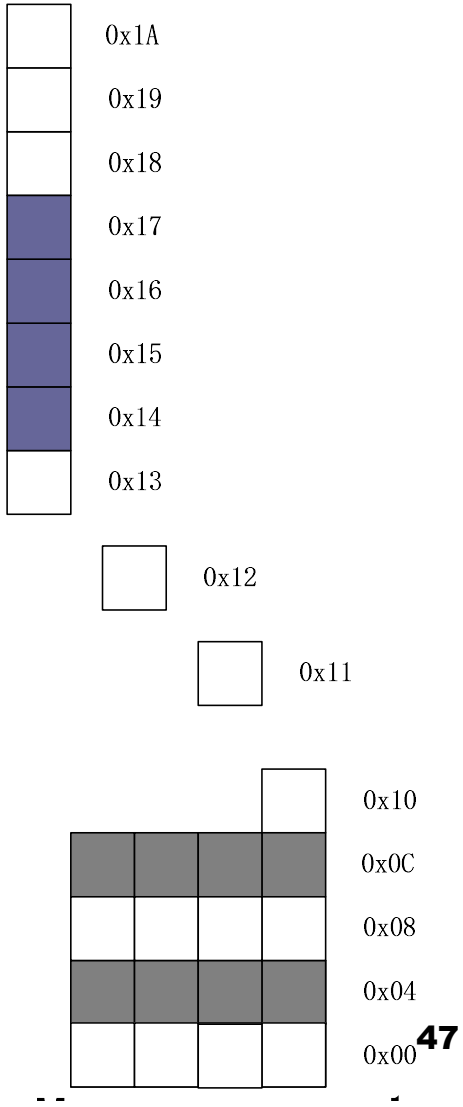
VI XỬ LÝ ARM7TDMI

- MMU interface
  - \trans (translation control), 0: user mode, 1: privileged mode
  - \mode[4:0], bottom 5 bits of the CPSR (inverted)
  - Abort, disallow access
- State
  - T bit, whether the processor is currently executing ARM or Thumb instructions
- Configuration
  - Bigend, big-endian or little-endian

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

VI XỬ LÝ ARM7TDMI

- Memory is addressed as a 32 bit address space
- Data type can be 8 bit bytes, 16 bit half-words or 32 bit words, and may be seen as a byte line folded into 4-byte words
- Words must be aligned to 4 byte boundaries, and half-words to 2 byte boundaries.
- Always ensure that memory controller supports all three access sizes



CHƯƠNG 2-BỘ VI XỬ LÝ ARM

VI XỬ LÝ ARM7TDMI

- Interrupt
  - \fiq, fast interrupt request, higher priority
  - \irq, normal interrupt request
  - isync, allow the interrupt synchronizer to be passed
- Initialization
  - \reset, starts the processor from a known state, executing from address 00000000_{16}

DÒNG CHẢY LỆNH

- Thời gian thực hiện chương trình:

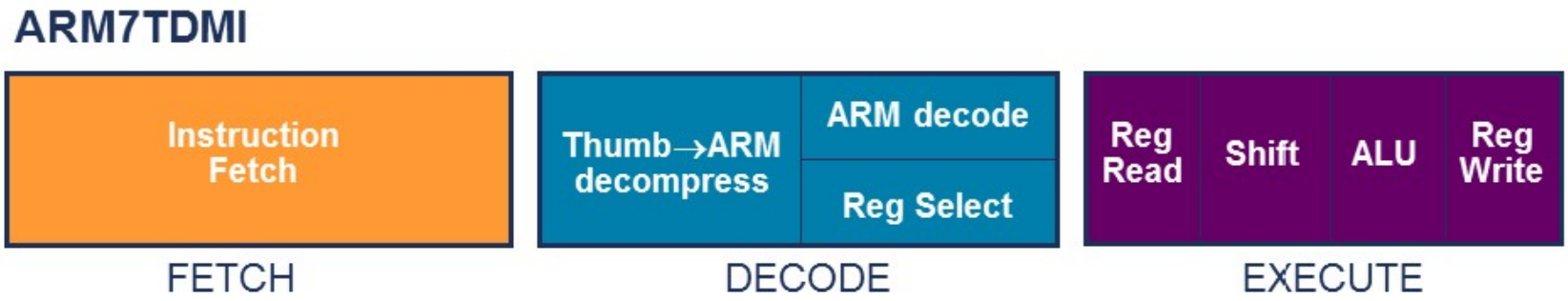
$$T_{prog} = \frac{N_{inst} \cdot CPI}{f_{clk}}$$

- Các cách để giảm thời gian thực hiện::
  - Tăng f_{clk}: ARM7 (66MHz), ARM9 (200MHz)
  - Giảm CPI: ARM7 (1.9), ARM9 (1.5)

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

DÒNG CHẢY LỆNH (INSTRUCTION PIPELINE)

- Arm7 sử dụng dòng chảy 3 tác vụ để tăng tốc độ xử lý:
  - Cho phép nhiều việc được thực hiện đồng thời thay vì thực hiện một cách tuần tự.

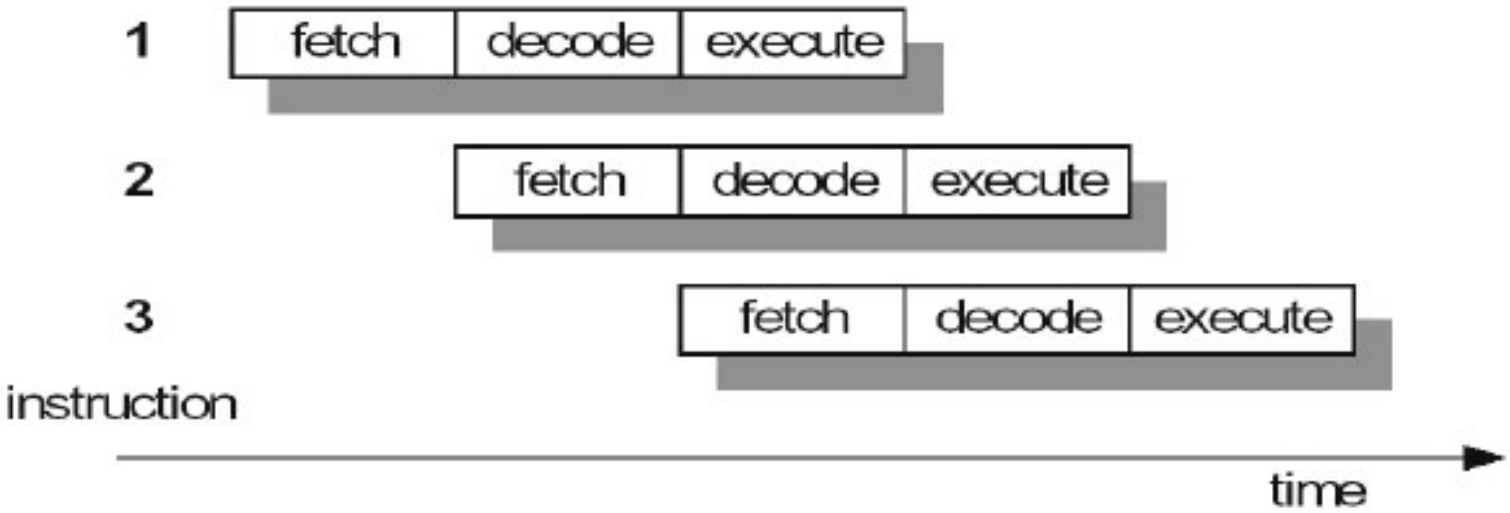


  - FETCH: Nạp lệnh từ bộ nhớ
  - DECODE: Giải mã lệnh
  - EXECUTE:
    - Đọc dữ liệu từ thanh ghi
    - Dịch chuyển số học và thực hiện tính toán
    - Ghi kế quả ngược trở lại thanh ghi

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

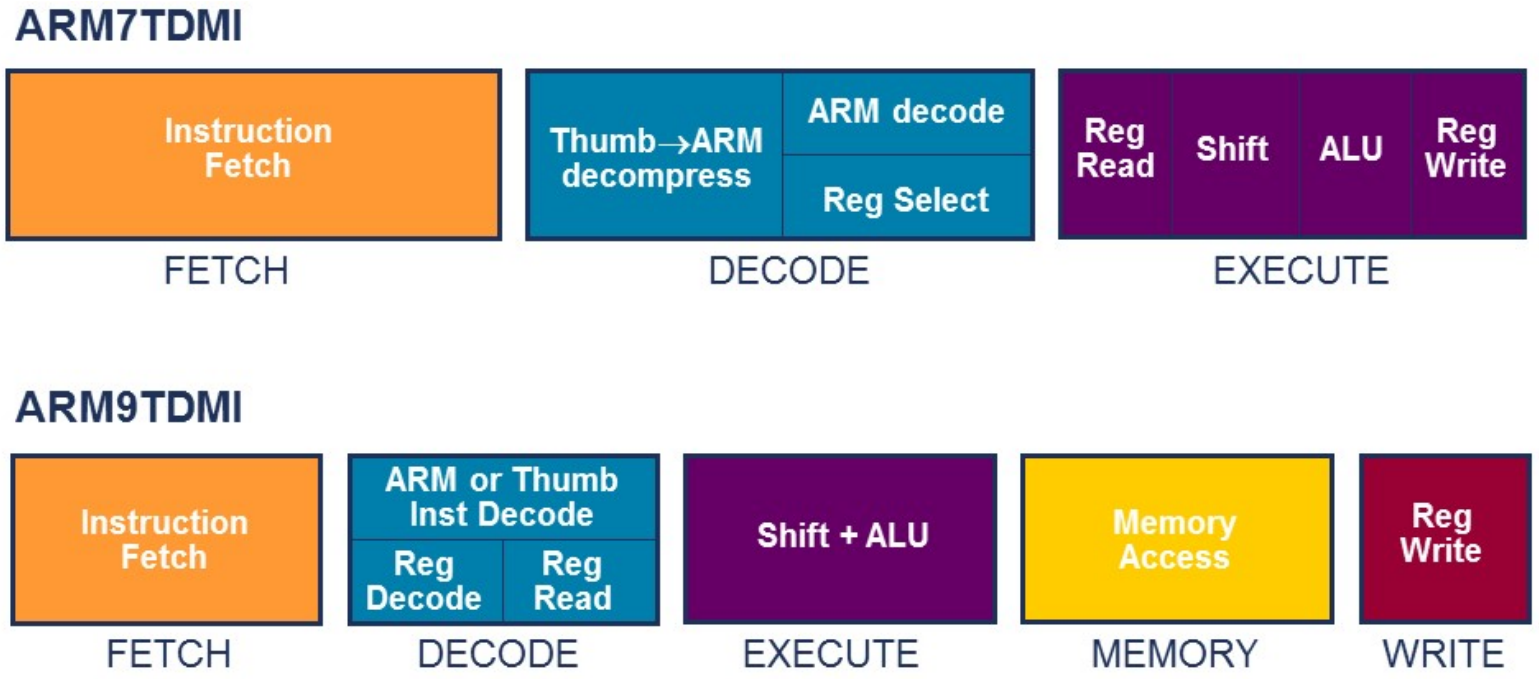
DÒNG CHẢY LỆNH (INSTRUCTION PIPELINE)

- Arm7 sử dụng dòng chảy 3 tác vụ để tăng tốc độ xử lý:



CHƯƠNG 2-BỘ VI XỬ LÝ ARM

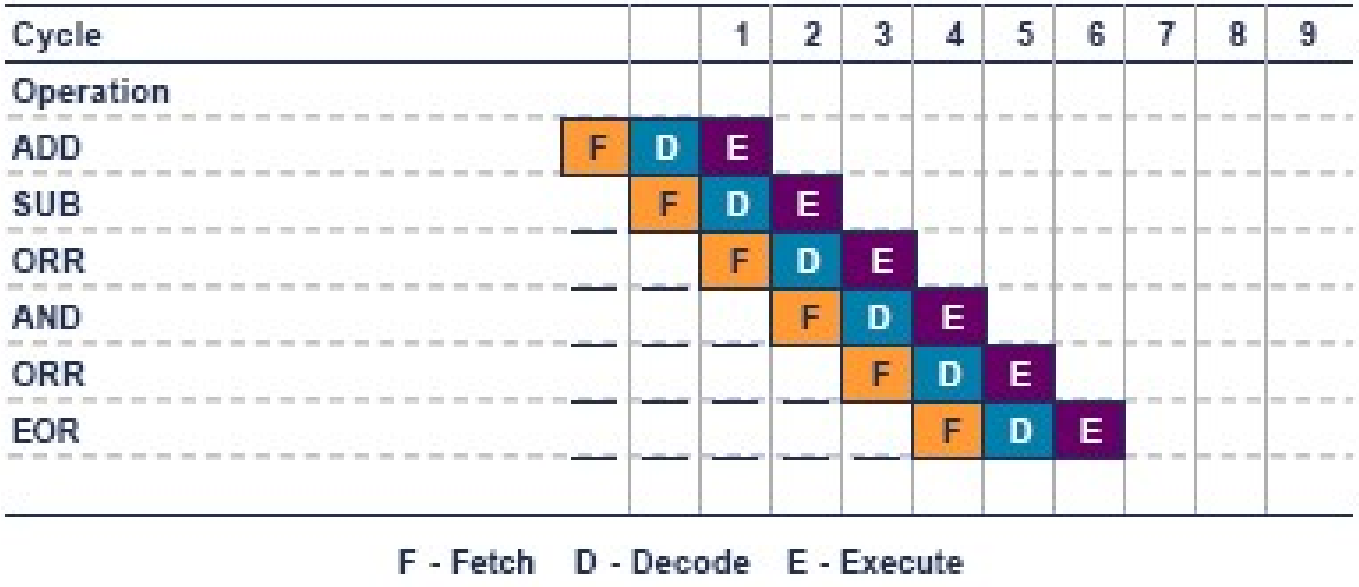
DÒNG CHẢY TÁC VỤ



- Arm7 thực hiện nhiều việc trong một tác vụ của dòng chảy lệnh.

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

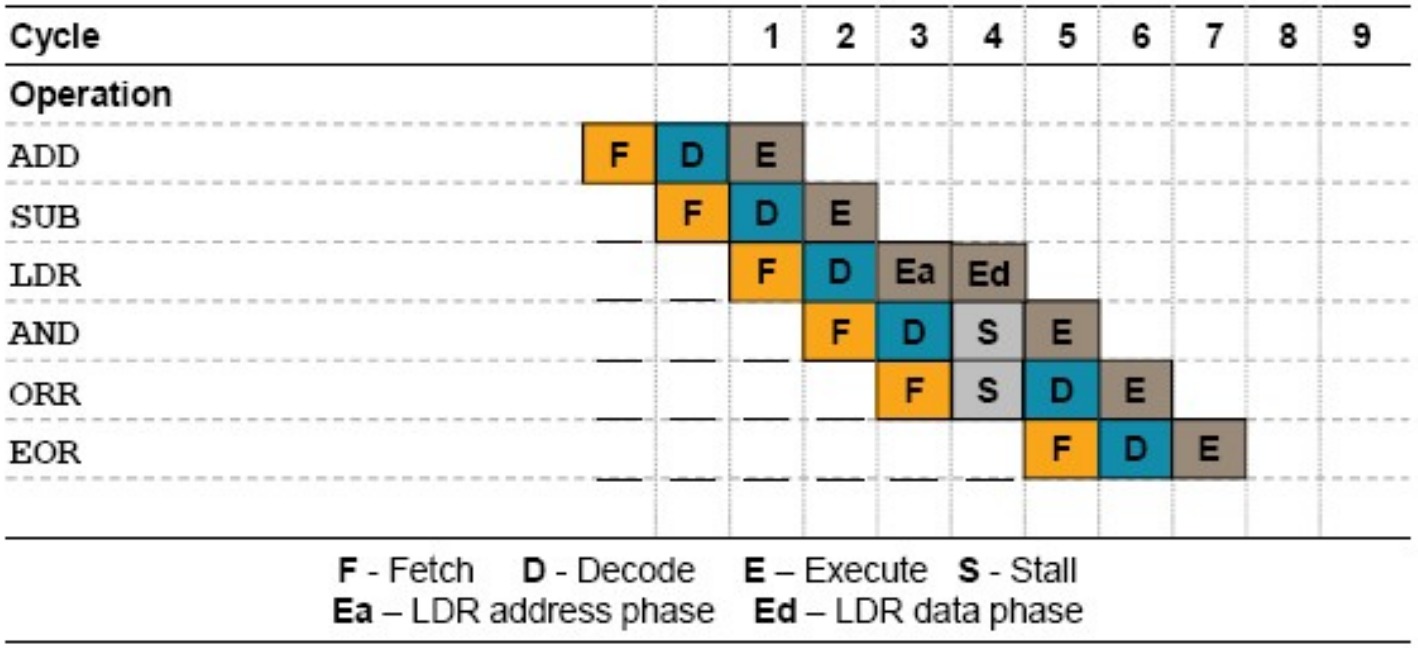
DÒNG CHẢY TÁC VỤ



- Các toán hạng đều là thanh ghi nên các lệnh được thực hiện trong 1 chu kỳ đơn.
- Cần 6 chu kỳ để thực hiện 6 lệnh => CPI = 1

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

DÒNG CHẢY TÁC VỤ

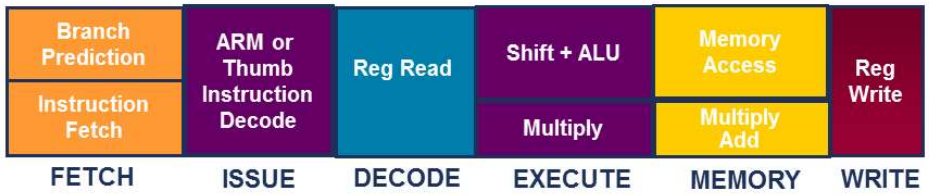


- Cần 7 chu kỳ để thực hiện 6 lệnh => CPI = 1.2

CHƯƠNG 2-BỘ VI XỬ LÝ ARM

DÒNG CHẢY TÁC VỤ

ARM10



ARM11