Interrupt e Direct Memory Access

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Architettura degli Elaboratori

Interrupt

Introduzione al concetto di Interrupt

Interrupt

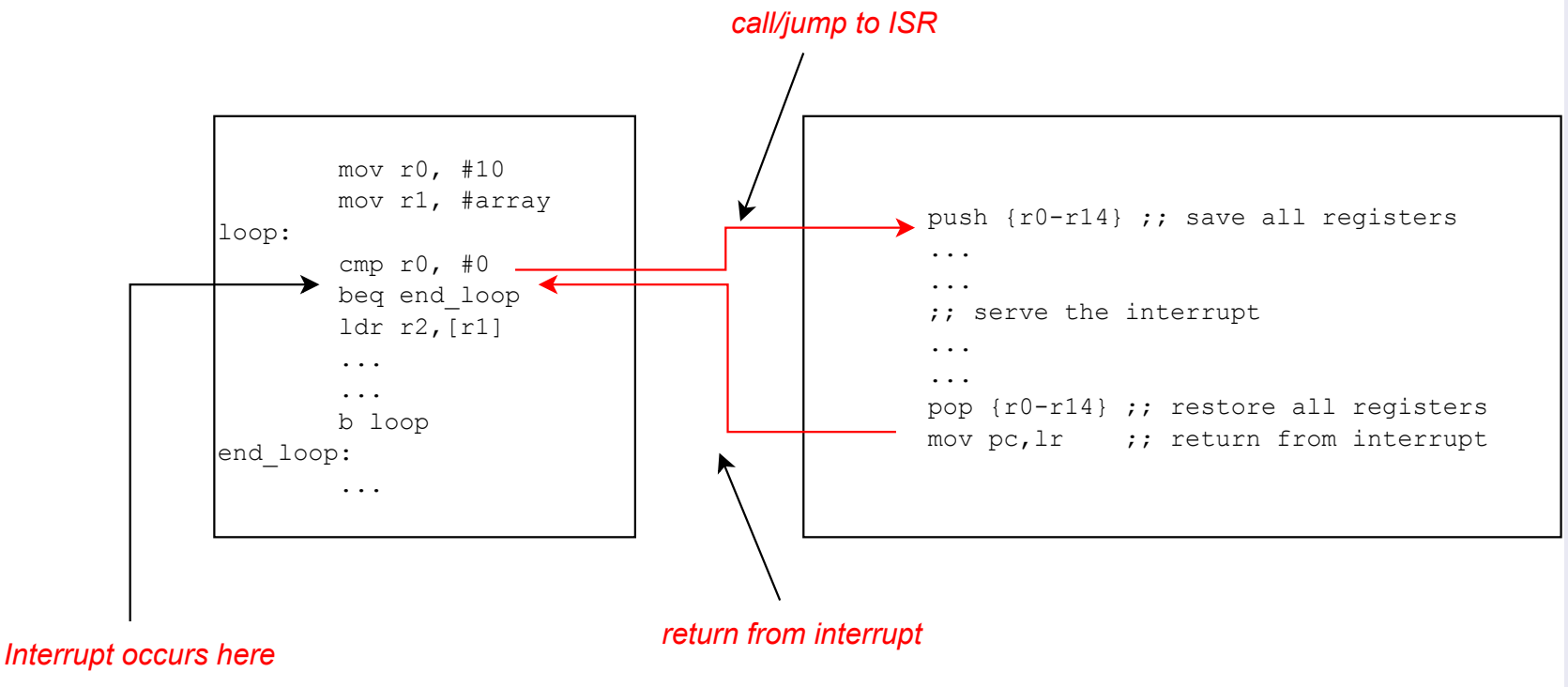
- La CPU esegue sempre un programma fatto di istruzioni poste *sequenzialmente* in memoria
- Durante l'esecuzione può accadere che accadano eventi dall'esterno che richiedono "attenzione"
  - pressione di un tasto della tastiera
  - arrivo di un pacchetto dall'interfaccia di rete
  - evento su USB (es. inserimento di un pendrive)
- Questi eventi sono inizialmente gestiti dall'hardware che "attiva" uno o più bit nei registri della relativa periferica di I/O
- E' compito poi del software andare a "interrogare" (**poll**) le locazioni di memoria di quei registri per controllare se e quale evento si sia verificato
- Tuttavia normalmente il software "fa altro", e non sta sempre lì a verificare gli eventuali eventi esterni....

Introduzione al concetto di Interrupt

Interrupt

- Il meccanismo degli Interrupt consente di svincolare il software dal dover interrogare (periodicamente o continuamente) i registri delle periferiche di I/O al fine di controllare se si è verificato un evento
- Quando si verifica un evento, l'hardware della periferica attiva i relativi bit nei registri (della periferica) e **invia un segnale di interruzione (interrupt) alla CPU** attraverso una linea specifica del control BUS
- La CPU **interrompe** l'esecuzione del programma corrente ed effettua un `JUMP` ad un indirizzo ben preciso (e prestabilito) dove deve trovarsi la routine che gestirà la richiesta di interruzione (**ISR - Interrupt Service Routine**)
- La ISR termina sempre con un'istruzione **Return-from-Interrupt** che indica alla CPU di riprendere l'esecuzione del programma interrotto precedentemente

Interrupt e ISR su ARM



Identificazione e dell'Interrupt

- Ogni CPU ha un ingresso di interrupt, generalmente denominato $\overline{INT}$ o $\overline{IRQ}$
- Quando una periferica intende generare un interrupt, essa pone questa linea allo stato logico 0 (oppure fronte di discesa)
- Il circuito di controllo della CPU identifica l'interrupt **appena prima** della fase di fetch
- Se l'ingresso di interrupt è a 0 la CPU attiva la procedura di **gestione dell'interrupt**

Gestione dell'Interrupt—senza Interrupt Controller

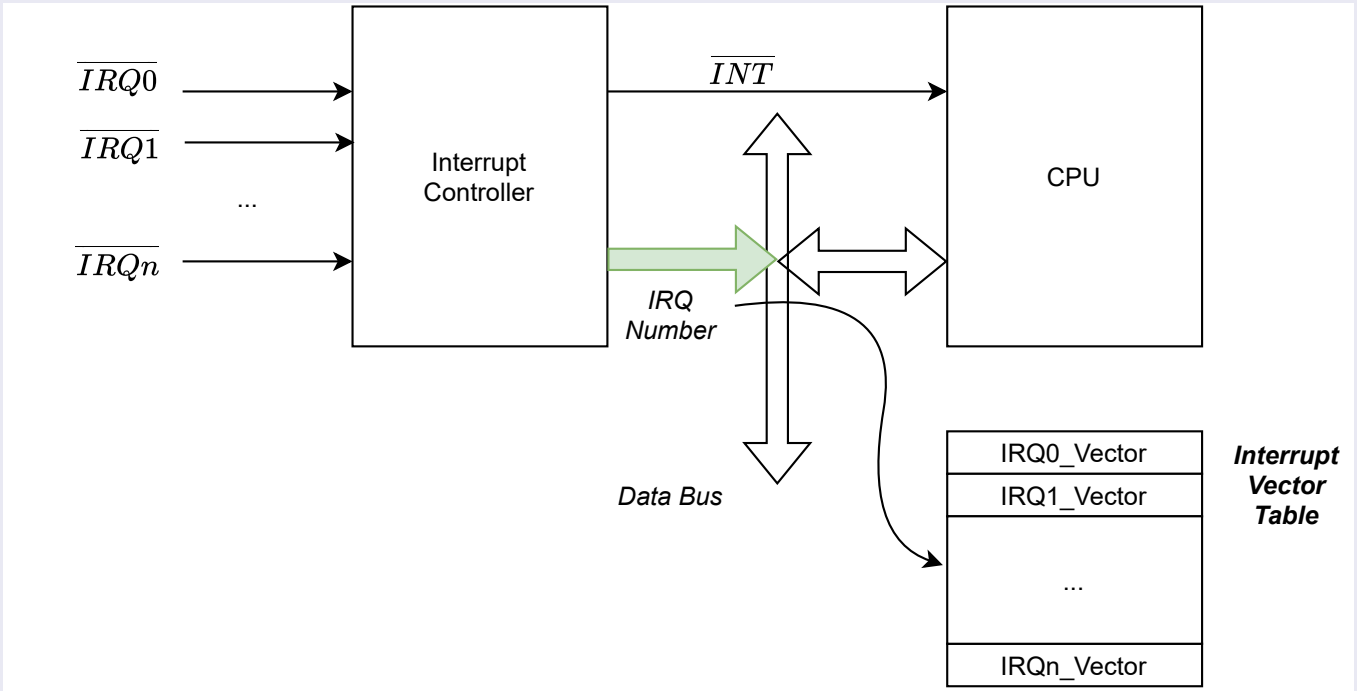
- La CPU salva l'attuale program counter nello stack, carica il program counter con un **indirizzo predefinito (Interrupt Vector)** e continua con il suo normale funzionamento
- All'Interrupt Vector deve essere presente la Interrupt Service Routine la quale:
  1. Salva sullo stack i registri che utilizzerà (oppure tutti)
  2. Esegue le operazioni relative alla gestione dell'evento
  3. Segnala alla CPU che l'interrupt è stato servito
  4. Recupera i registri dallo stack
  5. Esegue una "Return-from-Interrupt"

Gestione dell'Interrupt—con Interrupt Controller

- La CPU ha (in genere) un solo ingresso di Interrupt, ma le periferiche posso generare interrupt sono svariate, occorre pertanto un meccanismo di **multiplexing** che
  1. permetta di collegare diverse sorgenti di interrupt alla stessa linea di $\overline{INT}$
  2. consenta alla CPU di capire quale periferica ha richiesto attenzione
- Questo lavoro è svolto da un circuito esterno alla CPU denominato **Interrupt Controller** il quale lavora sempre accoppiato con la CPU specifica

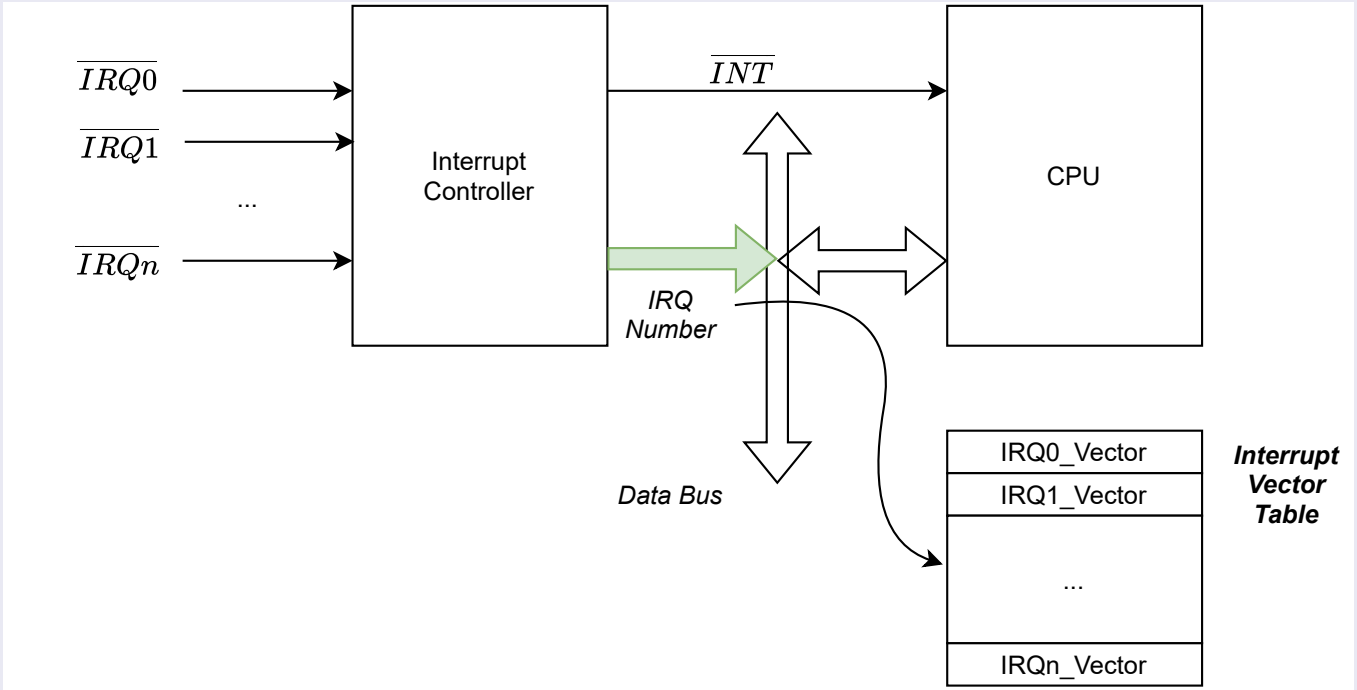
Gestione dell'Interrupt—con Interrupt Controller

- L'Interrupt Controller possiede diverse linee di ingresso di interrupt, denominate $\overline{IRQ0}$, $\overline{IRQ1}$, etc., e un'unica linea di uscita che si collega all'ingresso di interrupt della CPU
- Ogni linea di IRQ è connessa ad una periferica specifica



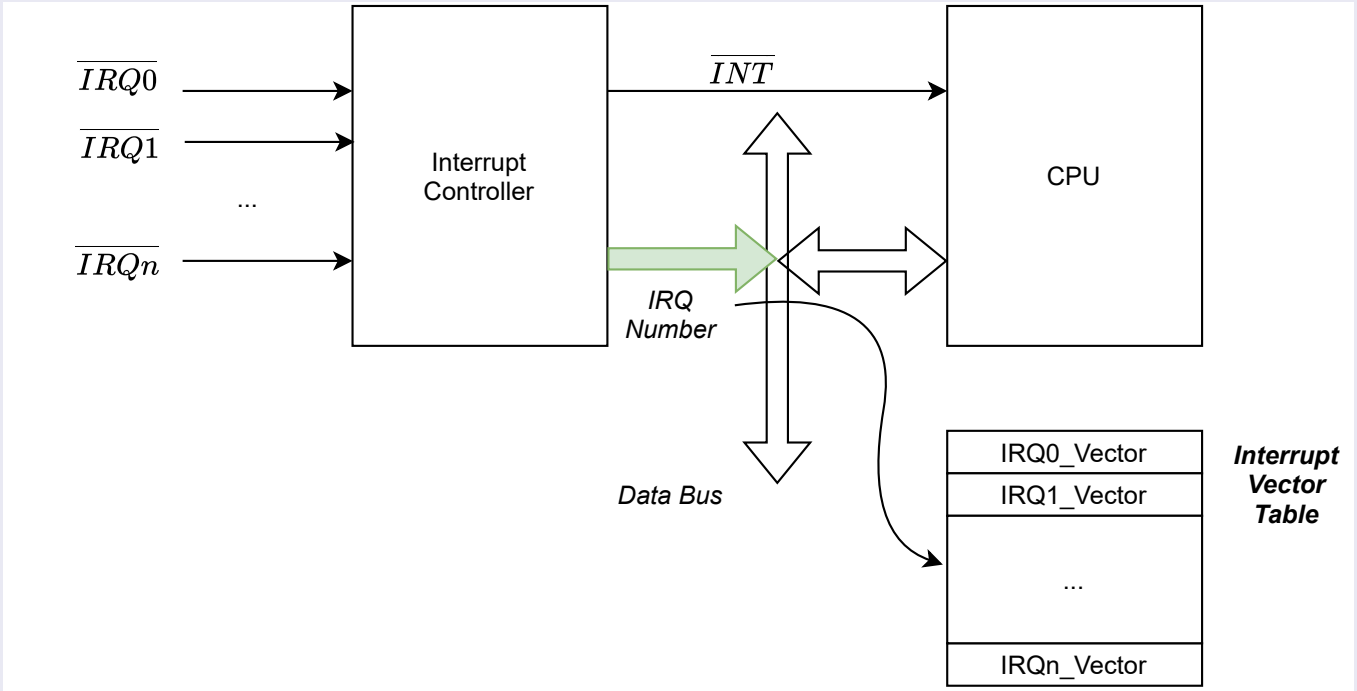
Gestione dell'Interrupt—con Interrupt Controller

- Quando un interrupt viene generato su una linea di IRQ, l'IC **memorizza (in un suo registro interno) il numero di IRQ e gira** la richiesta all'ingresso di interrupt della CPU



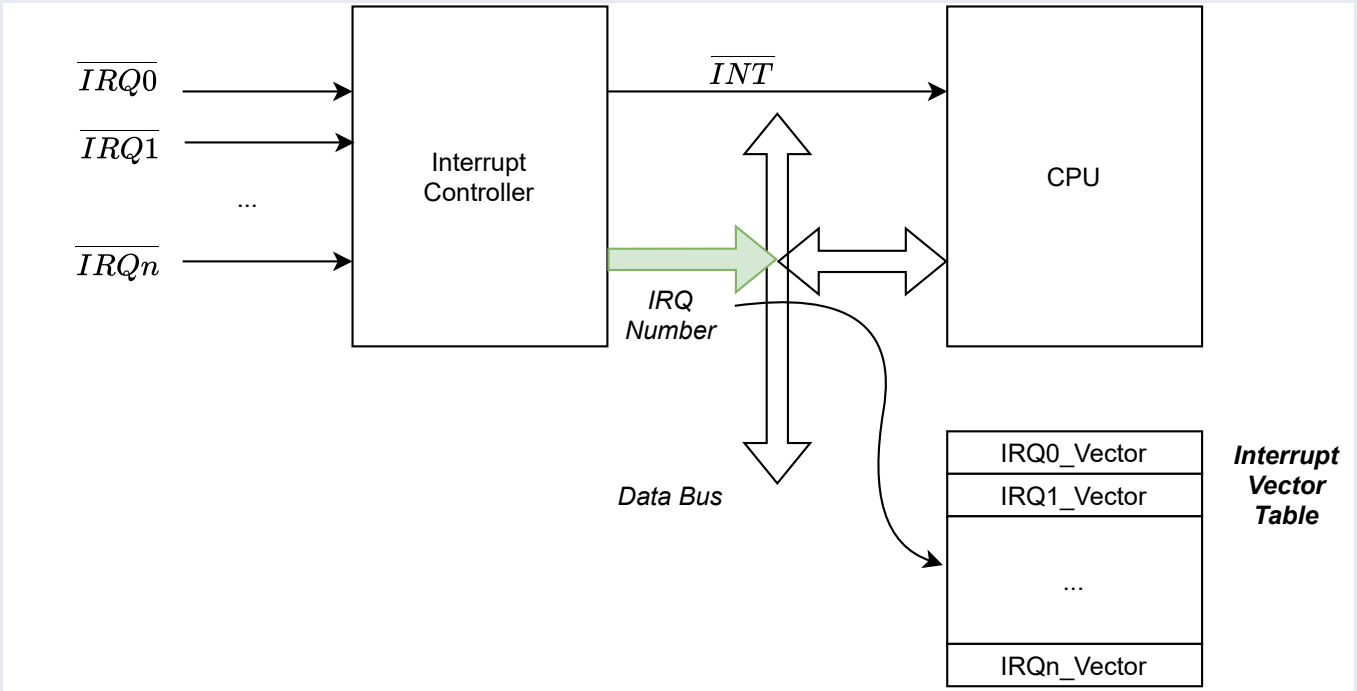
Gestione dell'Interrupt—con Interrupt Controller

- La CPU, dopo aver identificato l'evento, esegue una fase di **fetch** che non coinvolge la memoria, ma l'IC
- Durante tale fase, infatti, l'IC pone sul Data BUS il **numero dell'IRQ che era stato attivato**



Gestione dell'Interrupt—con Interrupt Controller

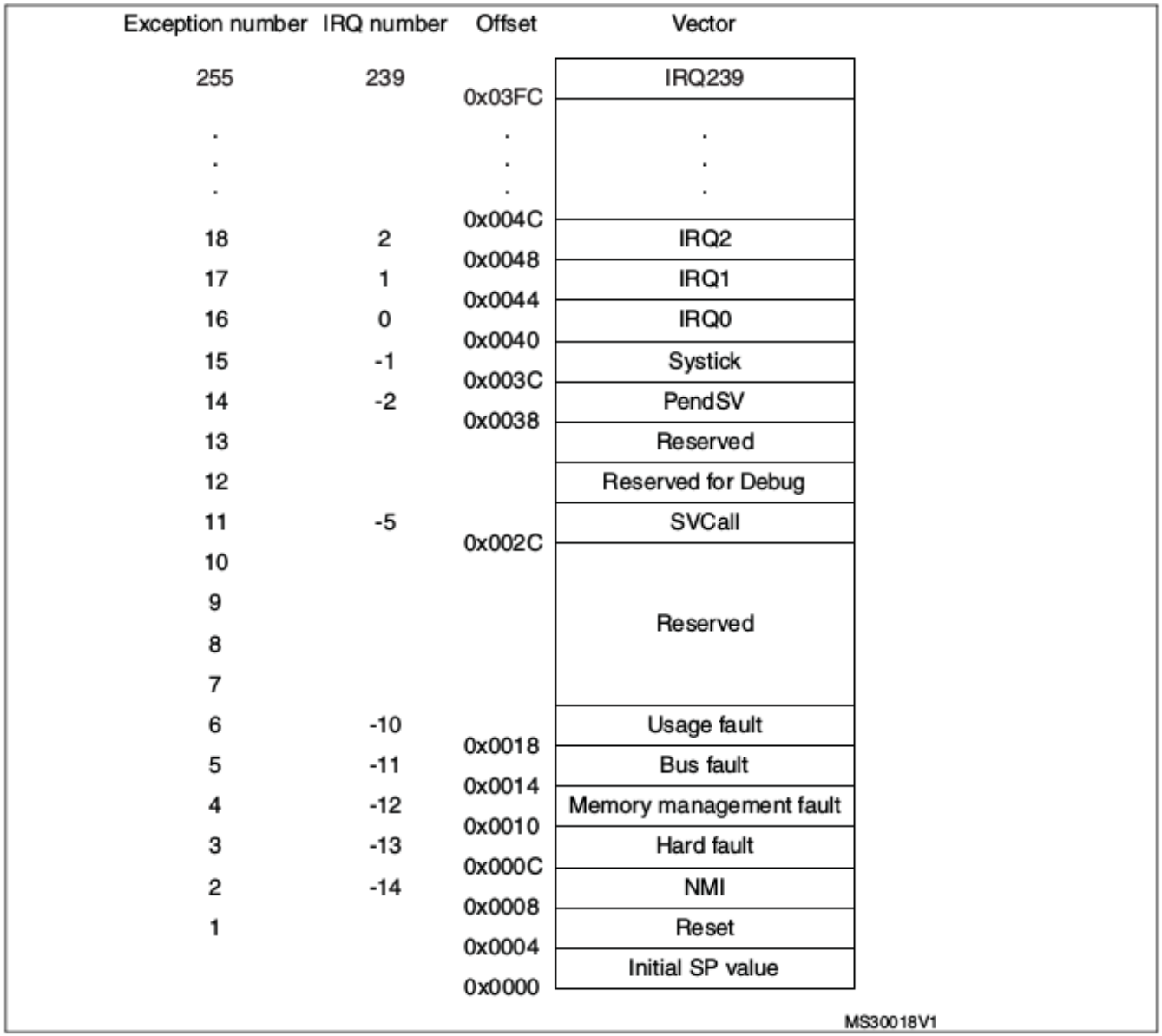
- La CPU acquisisce l'IRQ number e lo usa come offset di un tabella di **puntatori** denominata **Interrupt Vector Table**
- L'elemento indicizzato viene usato come **Interrupt Vector** e pertanto viene eseguito il salto all'indirizzo specificato



Exception

- Le **Exception (Eccezioni)** sono degli eventi analoghi agli Interrupt ma generati **internamente** dalla CPU
- Essi avvengono in seguito ad **eventi inaspettati (eccezionali)** che possono accadere durante l'esecuzione di un programma:
  - Istruzione illegale (opcode non riconosciuto)
  - Divisione per zero
  - Errore di Floating Point
  - Violazione di accesso alla memoria
  - ...
- A seguito di uno di questi eventi, la CPU interrompe il programma ed effettua un salto alla routine di servizio associata all'evento specifico
- Le entries della **IVT** contengono pertanto non solo i vettori degli interrupt esterni ma anche quelli delle **eccezioni**

Interrupt Vector Table, ARM Processor



Exception number	IRQ number	Offset	Vector
255	239	0x03FC	IRQ239
.		.	.
.		.	.
.		.	.
18	2	0x004C	IRQ2
17	1	0x0048	IRQ1
16	0	0x0044	IRQ0
15	-1	0x0040	Systick
14	-2	0x003C	PendSV
13		0x0038	Reserved
12			Reserved for Debug
11	-5		SVCall
10		0x002C	Reserved
9			
8			
7			
6	-10		Usage fault
5	-11	0x0018	Bus fault
4	-12	0x0014	Memory management fault
3	-13	0x0010	Hard fault
2	-14	0x000C	NMI
1		0x0008	Reset
		0x0004	Initial SP value
		0x0000	

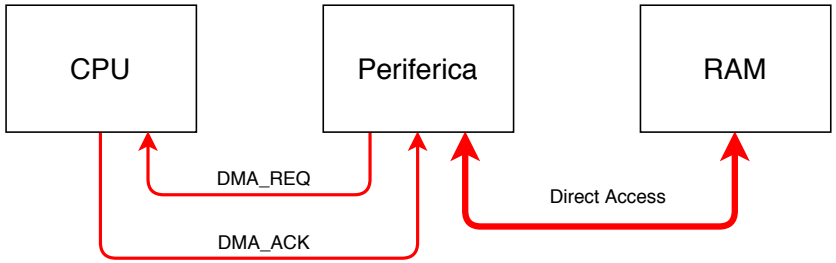
MS30018V1

Direct Memory Access

Direct Memory Access (DMA)

- Alcune periferiche hanno la necessità di trasferire una quantità di dati molto più grande di una singola word
- Ad esempio, un'interfaccia di memoria di massa (hard disk) trasferisce un **intero blocco** di disco, il quale equivale a 512 byte
- Poichè il trasferimento word-by-word, gestito dal software e dalla CPU potrebbe essere lento, si adotta una tecnica detta **Direct Memory Access (DMA)**
- Nel DMA, la periferica ha la possibilità di accedere direttamente alla memoria di sistema prendendone il controllo
- E' dunque possibile che la periferica "depositi" o "prelevi" un intero blocco di dati dalla RAM, senza passare dal controllo della CPU

Direct Memory Access



- Il Control Bus possiede due linee di handshake: *DMA_REQ* e *DMA_ACK*
- Quando una periferica vuole iniziare un trasferimento DMA, “attiva” la linea *DMA_REQ*
- La CPU conclude l’esecuzione dell’istruzione in corso e concede il Bus attivando, come risposta, la linea *DMA_ACK*
- La periferica riconose la riposta, prende il controllo del Bus ed effettua il trasferimento, al termine del quale “disattiva” la linea *DMA_REQ*, segnalando alla CPU che l’attività si è conclusa
- Opzionalmente, la periferica può generare un segnale di *IRQ* per far sì che il software possa gestire altre attività legate alla conclusione del trasferimento

Interrupt e Direct Memory Access

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Architettura degli Elaboratori