Architettura ARM

Da Wikipedia, l'enciclopedia libera.

Questa voce o sezione sull'argomento elettronica non cita le fonti necessarie o quelle presenti sono insufficienti.

L'**architettura ARM** (precedentemente **Advanced RISC Machine**, prima ancora **Acorn RISC Machine**), in elettronica e informatica, indica una famiglia di microprocessori RISC a 32-bit sviluppata da ARM Holdings e utilizzata in una moltitudine di sistemi embedded. Grazie alle sue caratteristiche di basso consumo elettrico rapportato alle prestazioni l'architettura ARM domina il settore dei dispositivi mobili dove il risparmio energetico delle batterie è fondamentale.

Logo dell'architettura ARM

Al 2007 la famiglia ARM copriva il 75% del mercato mondiale dei processori a 32 bit per applicazioni embedded,[1] posizionandosi come una delle più diffuse architetture a 32 bit del mondo. I processori ARM vengono utilizzati in cellulari, tablet, lettori multimediali, videogiochi portatili, PDA e periferiche per computer (come router, hard disk di rete ecc). Importanti rami della famiglia ARM sono i processori XScale e i processori OMAP, prodotti da Texas Instruments.

Indice

Storia

Il progetto ARM è iniziato nel 1983 nella sezione ricerca e sviluppo della Acorn Computers Ltd.

Un processore ARM Conexant utilizzato da molti router

Il team guidato da Sophie Wilson e Steve Furber iniziò lo sviluppo puntando a realizzare una versione migliore del MOS Technology 6502. Acorn in quel periodo utilizzava il MOS 6502 per i suoi computer e i manager pensarono che utilizzare processori prodotti internamente avrebbe fornito notevoli vantaggi.

Il team completò lo sviluppo del prototipo, chiamato semplicemente **ARM1**, nel 1985 e il primo processore realmente prodotto l'**ARM2** fu realizzato l'anno successivo. L'ARM2 era un processore con un bus dati a 32 bit, un bus di indirizzi a 26 bit, in modo da poter indirizzare fino a 64 Megabyte, e registri a 16/32 bit. Uno dei registri serviva per definire l'allineamento del program counter dato che i primi 6 bit e gli ultimi 2 erano utilizzati come flag dal processore per specificarne lo stato. L'ARM2 era probabilmente il più semplice processore a 32 bit con i suoi 30.000 transistor. Era più semplice del

Motorola 68000 che, pur avendo 68.000 transistor, forniva prestazioni paragonabili. Il processore doveva la sua semplicità alla mancanza di microcodice (nel 68000 il microcodice occupa un quarto dei transistor) e, come la maggior parte delle CPU dell'epoca, non aveva cache. Il successore ARM3 invece fu dotato di 4KB di cache per migliorare le prestazioni.

Alla fine degli anni ottanta Apple computer iniziò a lavorare con Acorn per sviluppare una nuova versione del core ARM. Il progetto era talmente importante da spingere Acorn a spostare il team di sviluppo in una nuova compagnia chiamata Advanced RISC Machines Ltd.. Per questo motivo spesso ARM è espanso come **Advanced RISC Machine** invece di Acorn RISC Machine. Advanced RISC Machines divenne ARM Ltd quando nel 1998 la società madre ARM Holdings decise di quotarsi al London Stock Exchange e al NASDAQ.

I lavori del team di sviluppo portarono alla realizzazione dell'**ARM6**. Il primo modello fu prodotto nel 1991 e Apple utilizzò il processore ARM 610 basato su core ARM6 per l'Apple Newton. Nel 1994 Acorn utilizzò ARM610 come processore centrale del suo computer RiscPC.

Il core era quasi immutato rispetto all'originale ARM2: mentre l'ARM2 aveva 30.000 transistor l'ARM6 ne aveva 35.000. L'idea della società infatti era di permettere agli OEM di prendere il core ARM e combinarlo con vari componenti opzionali per ottenere una cpu completa economica e a basso consumo, dato che era ottimizzata per i singoli compiti. Gli OEM poi avrebbero fatto produrre il processore a uno dei vari produttori di semiconduttori che lavorano su commissione.

L'implementazione di maggior successo è sicuramente l'ARM7TDMI, utilizzato in console portatili, telefoni cellulari e periferiche varie. Il processore è stato prodotto in centinaia di milioni di esemplari.

La DEC licenziò l'architettura ARM, producendo molta confusione nel mercato dato che la stessa DEC disponeva di una propria linea di processori i DEC Alpha, e produsse lo StrongARM. A 233 MHz il processore consuma solo 1 Watt e le versioni recenti consumano anche di meno. Intel, nel corso di una transazione con DEC per una causa legale, acquisì la linea StrongARM e la utilizzò in congiunzione con il suo processore i960 per realizzare la linea XScale che in seguito fu venduta durante una ristrutturazione aziendale.

L'architettura comunemente supportata da Windows Mobile e Android (sistemi operativi installati su PDA, smartphone, Tablet computer e altri dispositivi portatili) è l'ARM6. I processori XScale e l'ARM926 sono basati su **ARMv5TE** mentre gli StrongARM, gli ARM925T e gli ARM7TDMI sono basati su ARM4.

I core

Specifiche microprocessori ARM

Famiglia	Architettura	Core	Caratteristiche	Cache (I/D)/MMU	MIPS @ MHz	Applicazioni
ARM7TDMI	v4T	ARM7TDMI(-S)	3-stage pipeline	none	15 MIPS @ 16.8 MHz	Game Boy Advance, Nintendo DS, iPod
		ARM710T		MMU	36 MIPS @ 40 MHz	Psion 5 series, eMate 300
		ARM720T		8KB unified, MMU	60 MIPS @ 59.8 MHz	
		ARM740T		MPU		
	v5TEJ	ARM7EJ-S	Jazelle DBX	none		
ARM9TDMI	v4T	ARM9TDMI	5-stage pipeline	none		
		ARM920T		16KB/16KB, MMU	200 MIPS @ 180 MHz	Armadillo, Neo FreeRunner, GP32, GP2X (first core), Tapwave Zodiac (Motorola i. MX1)
		ARM922T		8KB/8KB, MMU		
		ARM940T		4KB/4KB, MPU		GP2X (second core)
ARM9E	v5TE	ARM946E-S		variabile, strettamente accorpata, MPU		Nintendo DS, Nintendo DSi, Nokia N-Gage, Conexant 802.11 chips, Samsung YP-K5 MP3 player
		ARM966E-S		no cache, TCMs		ST Micro STR91xF, incluso Ethernet [1] (https://web.archive.org/web/20070210162600/http://mcu.st.com/mcu/modules.php?

						name=mcu&file=devicedocs&DEV=STR912FW44&FAM=101)
		ARM968E-S		no cache, TCMs		
	v5TEJ	ARM926EJ-S	Jazelle DBX	variabile, TCMs, MMU	220 MIPS @ 200-220 MHz	Texas Instruments OMAP 16xx, 17xx; Nokia 6630, 6680, N70; Sony Ericsson (K, W series), Siemens and Benq (x65 series and newer), Caanoo
	v5TE	ARM996HS		no caches, TCMs, MPU		
ARM10E	v5TE	ARM1020E	(VFP)	32KB/32KB, MMU		
		ARM1022E	(VFP)	16KB/16KB, MMU		
	v5TEJ	ARM1026EJ-S	Jazelle DBX	variabile, MMU or MPU		
ARM11	v6	ARM1136J(F)-S	SIMD, Jazelle DBX, (VFP)	variable, MMU	?? @ 333-665 MHz (i.MX31 SoC)	Texas Instruments OMAP2, Nokia E51 (369 MHz), Nokia 6700 Classic, Nokia 6120 classic, Nokia 6220 Classic (369 MHz), Nokia 6720 Classic, Nokia 6290, Nokia 6210 Navigator, Nokia 6710 Navigator, Nokia N93, Nokia N95 (333 MHz), Nokia Nst-4 (434 MHz), Nokia E5 (600 MHz), Nokia 5800 Xpressmusic (434 MHz), Nokia N97 (434 MHz) HTC Dream, HTC Magic, HTC Hero
	v6T2	ARM1156T2(F)-S	SIMD, Thumb-2, (VFP)	variabile, MPU		
	v6KZ	ARM1176JZ(F)-S	SIMD, Jazelle DBX, (VFP)	variabile, MMU+TrustZone		Raspberry Pi, iPhone EDGE, iPhone 3G
	v6K	ARM11 MPCore	1-4 core SMP, SIMD, Jazelle DBX, (VFP)	variabile, MMU		
Cortex	v7-A	Cortex-A8	NEON, Jazelle RCT, Thumb-2	variabile (L1+L2), MMU+TrustZone	up to 2000 (2.0 DMIPS/MHz in speed from 600 MHz to greater than 1 GHz)	Texas Instruments OMAP3, Freescale famiglia i.MX51 [2] (http://www.freescale.com/webapp/sps/site/taxonomy.jsp?code=IMX51_FAMILY)
	v7-R	Cortex-R4	Embedded	cache variabile, MMU opzionale	600 DMIPS	Broadcom
	v6-M	Cortex-M0	Microcontroller	with cache, senza MMU	fino a 50 MHz	NXP famiglia LPC11xx [3] (http://ics.nxp.com/lpczone/)
	v7-M	Cortex-M3	Microcontroller	no cache, (MPU)	120 DMIPS @ 100 MHz	Luminary Micro [4] (http://www.luminarymicro.com) microcontroller family, STMicroelectronics famiglia STM32 [5] (http://www.st.com/mcu/inchtml-pages-stm32.html), NXP famiglie LPC17xx e LPC13xx [6] (http://ics.nxp.com/lpczone/), ATMEL famiglia

						AT91SAM3x [7] (http://www.atmel.com/products/at91/sam3s.asp?family_id=605).
	v7E-M	ARM Cortex-M4	Microcontroller	MPU	1.25 DMIPS/MHz	Freescale Kinetis, NXP
XScale	v5TE	80200/IOP310/IOP315	I/O Processor			
		80219				
		IOP321				Iyonix
		IOP33x				
		PXA210/PXA250				Zaurus SL-5600, IPaq 54xx
		PXA255		32KB/32KB, MMU	400 BogoMips @400 MHz	Gumstix, IPaq 55xx
		PXA26x				
		PXA27x			800 MIPS @ 624 MHz	HTC Universal, Zaurus SL-C1000,3000,3100,3200
		PXA800(E)F				
		Monahans			1000 MIPS @ 1.25 GHz	
		PXA900				Blackberry 8700
		IXC1100	Control Plane Processor			
		IXP2400/IXP2800				
		IXP2850				
		IXP2325/IXP2350				
		IXP42x				NSLU2
		IXP460/IXP465				

Note di progetto

Per mantenere il progetto pulito, semplice e veloce il processore non ha microcodice, come il processore MOS 6502 utilizzato nei primi computer Acorn.

L'architettura ARM è un'architettura RISC che prevede:

- Architettura load/store
- assenza del supporto ad accessi non allineati alla memoria (supportati dal core v6)
- set di istruzioni ortogonale
- numerosi registri a 16/32 bit
- istruzioni a lunghezza fissa per semplificare la decodifica e l'esecuzione a costo di diminuire la densità del codice
- Completamento di un'istruzione per ogni ciclo di clock in una situazione puramente ideale priva di stalli

Per compensare il progetto semplice rispetto a processori della stessa epoca come l'Intel 80286 e il Motorola 68020, il processore comprendeva alcune caratteristiche uniche come:

- esecuzione condizionata di molte istruzioni per ridurre i salti e compensare gli stalli della pipeline
- solo le operazioni aritmetiche possono alterare i registri delle esecuzioni condizionate
- shifter a 32 bit che può essere utilizzato in contemporanea con la maggior parte delle istruzioni senza penalizzazioni di tempo
- metodo di indirizzamento a indice molto potente
- interrupt a 2 livelli molto veloce e semplice con un sottosistema di registri collegati che commutano

Una delle caratteristiche più interessante dei processori ARM sono 4 bit addizionali utilizzati per realizzare dei codici condizionali per ogni istruzione.

Questi codici hanno ridotto le possibilità di indirizzo dato che il processore non ha molti bit per poterli specificare, ma il grande vantaggio è che questi codici permettono di evitare i salti nei caso di semplici `if`. Un esempio classico si ha nell'esecuzione dell'algoritmo di Euclide per la ricerca del massimo comune divisore (MCD).

In linguaggio C il codice è:

```c
int gcd (int i, int j)
{
    while (i != j)
    {
       if (i > j)
           i -= j; /*in forma classica i=i-j;*/
       else
           j -= i; /*in forma classica j=j-i;*/
    }
    return i;
}
```

In assembly ARM il ciclo diventa:

```
loop   CMP    Ri, Rj       ; set condition "NE" if (i != j)
                            ;               "GT" if (i > j),
                            ;          or   "LT" if (i < j)
       SUBGT  Ri, Ri, Rj    ; if "GT", i = i-j;
       SUBLT  Rj, Rj, Ri    ; if "LT", j = j-i;
       BNE    loop          ; if "NE", then loop
```

Questo evita i rami del `then` e dell'`else`.

Un'altra caratteristica unica del set di istruzioni è la capacità di shiftare i dati durante le normali operazioni sui dati (operazioni aritmetiche, logiche e di copia di registri). Per esempio il codice C seguente

```
a += (j << 2);
```

viene tradotto in questa unica istruzione assembly eseguita in un solo ciclo

```
ADD Ra, Ra, Rj, LSL #2
```

Queste caratteristiche rendono i programmi ARM normalmente più *densi* degli equivalenti programmi per altri processori RISC. Inoltre il processore fa meno accessi alla memoria e riesce a riempire meglio le pipeline. Quindi le CPU ARM possono utilizzare frequenze inferiori a quelle di altri processori consumando meno potenza per svolgere gli stessi compiti.

Inoltre un processore ARM ha altre caratteristiche viste raramente nei processori RISC come l'indirizzamento relativo al PC (il PC negli ARM è un registro a 16 bit), indirizzamento con il pre e post incremento.

Una caratteristica *curiosa* dei processori ARM è che con il tempo il set di istruzioni incrementa. I primi processori ARM (prima dell'ARM7TDMI) per esempio non avevano istruzioni per caricare quantità a due byte. E quindi non era in grado di gestire direttamente i tipi *short* in C.

I primi processori tipo gli ARM7 erano basati su un disegno con pipeline a 3 stadi, fetch, decode e execute. I processori più moderni come l'ARM9 per incrementare le prestazioni sono passati a pipeline a 5 stadi. Altri cambiamenti per incrementare le prestazioni includono un sommatore veloce e un sistema di predizione dei salti.

Thumb

Gli ultimi processori ARM sono dotati di un set di istruzioni a 16 bit chiamato **Thumb** che utilizza quattro byte per ogni istruzione. Il codice Thumb è più leggero, ma è dotato di meno funzionalità. Per esempio solo i salti possono essere condizionati e alcuni opcode non possono essere utilizzati da tutte le istruzioni. Nonostante queste limitazioni, Thumb fornisce prestazioni migliori del set di istruzioni completo nel caso di sistemi dotati di limitata larghezza di banda. Molti sistemi embedded sono dotati di un bus verso la memoria limitato e, sebbene il processore possa indirizzare a 32 bit, spesso si utilizzano indirizzamenti a 16 bit o simili: un esempio molto diffuso è il Game Boy Advance. In queste situazioni conviene creare codice Thumb per la maggior parte del programma e ottimizzare le parti di codice che richiedono molta potenza di calcolo utilizzando il set di istruzioni completo.

Il primo processore dotato di Thumb è stato l'ARM7TDMI. Tutti gli ARM9 e le famiglie successive (incluso gli XScale) sono dotati di Thumb.

Jazelle

ARM ha implementato in alcuni processori la tecnologia[2] per permettere al processore di eseguire nativamente il Java bytecode. Questa tecnologia è interoperabile con il codice ARM standard e Thumb.

Il primo processore dotato di Jazelle è stato l'**ARM926J-S**, la tecnologia Jazelle è stata sottolineata dalla J nella sigla. Il processore viene utilizzato sui telefoni cellulari per velocizzare l'esecuzione dei giochi Java ME e delle applicazioni. L'idea di facilitare l'esecuzione di codice Java per queste applicazioni ha probabilmente spinto ARM a sviluppare questa tecnologia.

Thumb-2

La tecnologia **Thumb-2** ha fatto il suo debutto nell'**ARM1156 core**, presentato nel 2003. Thumb-2 estende le limitate istruzioni a 16 bit con delle addizionali istruzioni a 32 bit per fornire maggior potenza al processore. La tecnologia Thumb 2 fornisce codice con densità (e quindi occupazione di banda) simile a quello del codice Thumb ma con prestazioni più vicine a quello ARM a 32 bit.

Thumb-2 inoltre estende le istruzioni ARM e Thumb con delle nuove istruzioni che permettono la manipolazione dei singoli bit, l'esecuzione condizionata e la gestione di tabelle con salti.

Thumb-2EE

Thumb-2EE, venduta come JazelleRCT[3], è una tecnologia annunciata nel 2005 che è stata implementata per la prima volta nel processore **Cortex-A8**. Thumb-2EE è una estensione delle istruzioni Thumb-2, specificamente progettato per gestire codice generato in tempo reale, per esempio durante la esecuzione di codice compilato just in time. La tecnologia Thumb 2EE è stata progetta per linguaggi come Java, C#, Perl e python in modo da generare codice compilato di dimensioni ridotte senza impattare sulle prestazioni.

Le nuove istruzione fornite permettono di controllare automaticamente i puntatori nulli prima di ogni load o store, permettono di gestire l'eventuale sfondamento degli array, la gestione delle diramazioni e molte altre caratteristiche fornite da linguaggi ad alto livello come l'istanza di memoria per i nuovi oggetti.

NEON

La tecnologia **NEON** è una combinazioni di istruzioni a 64 e 128 bit SIMD (Single Instruction Multiple Data) per accelerare e standardizzare il trattamento e l'elaborazione di segnali multimediali. NEON permette di eseguire la decodifica MP3 con una CPU a 10 Megahertz e permette di eseguire il codec GSM AMR (Adaptive Multi-Rate) con una CPU a 13 Megahertz. La tecnologia poggia su un set di istruzioni separato, registri indipendenti e esecuzione del codice separata. NEON gestisce dati a 8/16/32/64 bit di tipo intero, a singola precisione e in virgola mobile. La tecnologia SIMD è cruciale per l'esecuzione di operazioni vettoriali, operazioni che trattano molti dati con lo stesso programma. NEON permette di gestire fino a 16 operazioni contemporaneamente.

VFP

La tecnologia **VFP** è un'estensione dell'architettura ARM che fornisce un coprocessore matematico. La tecnologia è nata per fornire operazioni in grado di trattare dati in virgola mobile a singola e doppia precisione in modo economico e pienamente compatibile con la standard *ANSI/IEEE Std 754-1985 Standard for Binary Floating-Point Arithmetic*. VFP fornisce istruzioni per applicazioni tipo compressioni, decompressioni, grafica tridimensionale, analisi audio e altro. Questa estensione risulta utile per dispositivi tipo PDA, smartphone, set-top box e applicazioni di automazione e controllo. La tecnologia VFP gestisce anche brevi vettori di dati con tecnologia SIMD.

Licenze ARM

ARM Ltd non produce realmente le sue CPU e non vende dispositivi basati sulle sue CPU. ARM Ltd licenzia ad altre aziende la possibilità di realizzare CPU basate su core ARM. ARM offre una serie di licenze che variano a seconda del processore, delle personalizzazioni e del numero di pezzi prodotti. Tutte le licenze ARM prevedono una descrizione hardware dei core e il set completo di strumenti di sviluppo per la realizzazione del software per i processori. Le aziende, dopo aver acquistato la licenza da ARM, si rivolgono a una fonderia di silicio che provvede a realizzare fisicamente il processore basandosi sulla descrizione fornita da ARM. Se l'acquirente lo richiede, ARM fornisce anche i simulatori dell'hardware delle CPU in modo da permettere all'acquirente di modificare la CPU aggiungendovi funzionalità e poi di testarne l'effettiva funzionalità senza dover realizzare fisicamente il chip. Alcuni acquirenti acquistano direttamente la descrizione verilog dei processori in modo da poter ottimizzare anche il singolo core migliorandone le prestazioni o riducendone i consumi per utilizzi particolari. Caso a parte sono le fonderie di silicio: queste possono effettuare delle modifiche per un cliente e le stesse modifiche possono essere vendute ad altri clienti.

Come le altre aziende, ARM vende le licenze in base al loro valore percepito. Core lenti vengono venduti a un prezzo inferiore a quello di core più moderni e veloci. Inoltre versioni ottimizzate hanno costi maggiori di versioni realizzate assemblando blocchi precostituiti. La situazione viene ulteriormente complicata dalla presenza di fonderie che rivendono versioni ridotte delle licenze ARM (Samsung e Fujitsu per esempio), avendo preso dei core base e avendoli personalizzati per migliorarne le prestazioni. Rispetto a fonderie che lavorano su

progetti dedicati e quindi realizzati espressamente, queste fonderie offrono prezzi 2 o 3 volte inferiori con prestazioni spesso accettabili. Nel caso di bassi volumi di acquisto usualmente conviene rivolgersi alle fonderie per le licenze mentre in caso di alti volumi di vendita conviene contattare direttamente ARM per una licenza specifica e poi rivolgersi a fonderie dedicate che lavorano solo per il committente e non come Samsung o Fujitsu che lavorando per più committenti contemporaneamente non possono seguire il prodotto allo stesso modo.

I maggiori gruppi di semiconduttori hanno licenza con ARM. Tra questi: Atmel, Broadcom, Cirrus Logic, Freescale (società spin-off di Motorola dal 2003), Fujitsu, Intel (tramite un accordo con DEC), IBM, Infineon Technologies, Nintendo, OKI, Philips, Samsung, Sharp, STMicroelectronics, Texas Instruments e VLSI. I contratti di licenza sono regolati da clausole di non divulgazione quindi non si conoscono con certezza i costi dei core ARM anche se si sa che sono tra i core più costosi del mercato a parità di prestazioni. Una singola licenza ARM per un prodotto con un processore ARM può costare fino a 200.000 dollari. Per quantità elevate e per modifiche significative all'architettura il costo della licenza può superare i 10 milioni di dollari.

Note

1. **^** *Copia archiviata* (PDF), su *arm.com*. URL consultato il 17 dicembre 2007 (archiviato dall'url originale il 3 dicembre 2007).
2. **^** Jazelle (http://www.arm.com/products/multimedia/java/jazelle.html) Archiviato (https://web.archive.org/web/20090223212504/http://www.arm.com/products/multimedia/java/jazelle.html) il 23 febbraio 2009 in Internet Archive.
3. **^** Jazelle RCT (http://www.arm.com/products/solutions/JazelleRCT.html)

Voci correlate

- AMULET
- Architettura Harvard
- ARM11
- Confronto tra microprocessori ARM

Altri progetti

- Wikimedia Commons (https://commons.wikimedia.org/wiki/?uselang=it) contiene immagini o altri file su **Architettura ARM (https://commons.wikimedia.org/wiki/Category:ARM_microprocessors?uselang=it)**

Collegamenti esterni

- (EN) *ARM Ltd.*, su *arm.com*.
- *Acorn RISC Machine: l'origine della specie*, su *appuntidigitali.it*.
- *Come funziona un processore ARM*, su *oscene.net*.
- *Le estensioni DSP all'architettura ARM*, su *appuntidigitali.it*.
- *Le estensioni SIMD dell'architettura ARM*, su *appuntidigitali.it*.

Controllo di autorità	LCCN (EN) sh2015001756 (http://id.loc.gov/authorities/subjects/sh2015001756) · GND (DE) 4706184-4 (https://d-nb.info/gnd/4706184-4) · BNF (FR) cb16243194b (https://catalogue.bnf.fr/ark:/12148/cb16243194b) (data) (https://data.bnf.fr/ark:/12148/cb16243194b)