

Sistemi Embedded

Microprocessori

Luigi Pomante
Università dell'Aquila – DEWS
luigi.pomante@univaq.it

Sommario

- Introduzione
 - Microprocessori general purpose
 - Architetture CISC, RISC, Superscalari
 - Architetture CISC/RISC, EPIC/VLIW, CISC/VLIW
 - Analisi comparata
 - Microprocessori dedicati
 - Digital Signal Processor
 - Network Processor
 - Microcontrollori
 - ...
 - Conclusioni

Microprocessori

Introduzione

Introduzione

- Molte applicazioni embedded realizzano parte delle loro funzionalità mediante algoritmi realizzati sotto forma sw
 - La principale motivazione è nella flessibilità tipica del sw rispetto alla inerente rigidità dei componenti hardware digitali
- In questo contesto il concetto di flessibilità assume diversi significati tra cui i principali riguardano la **manutenibilità** e l'**estendibilità**
 - Lo sviluppo di funzionalità in software risulta in generale meno complesso e richiede minori tempi di sviluppo
 - Anche la verifica, sebbene non semplice, risulta meno critica

Introduzione

- Esiste, come prevedibile, il rovescio della medaglia
 - Le prestazioni delle soluzioni SW risultano praticamente sempre peggiori dell'equivalente soluzione HW
 - Questo perchè le architetture di calcolo su cui il SW si appoggia, per quanto ottimizzate e specializzate, rimangono comunque molto generiche rispetto alla specificità di ogni singolo problema
 - Si tenga inoltre presente che per prestazioni non si intende solamente la potenza di calcolo
 - Dissipazione di potenza, memoria richiesta, area occupata dal chip e relativo package, ecc...

Introduzione

- Esempio (1/2)
 - Si consideri il problema della somma di due interi a 32 bit, e si confronti una soluzione realizzata mediante HW dedicato e una implementata in SW su un microprocessore a 32 bit
 - Si supponga di lavorare a parità di frequenza di clock
 - Nel primo caso, una istruzione assembly è sufficiente a risolvere il problema mentre per la soluzione hardware risulta necessario sviluppare l'intero sommatore
 - Se ne conclude che la soluzione software sembrerebbe preferibile

Introduzione

- Esempio (2/2)
 - Si immagini ora di passare da 32 a 33 bit perché, per esempio, si è valutato che la precisione necessaria è maggiore di quella inizialmente prevista
 - La soluzione hardware, benché richieda modifiche, rimane pressoché invariata in termini di architettura e di prestazioni
 - La soluzione software richiede due o, più verosimilmente, tre istruzioni assembly, con un peggioramento delle prestazioni dell'ordine del 100% circa

Sistemi Embedded
2010/2011

7

Introduzione

- L' esempio (estremizzato) evidenzia la relazione che esiste tra una soluzione HW dedicata e una soluzione SW più generica e flessibile
 - Optare per una soluzione SW richiede un'ottima conoscenza delle architetture di microprocessori e delle soluzioni disponibili sul mercato
 - A guidare il progettista nella scelta devono essere le caratteristiche del problema (algoritmo) e l'insieme dei vincoli (prestazioni, costi, tempi di sviluppo e così via) cui è sottoposto
 - Prima ancora della scelta di uno specifico microprocessore, il problema da affrontare riguarda la scelta della classe di processore e della forma in cui lo si desidera acquisire e utilizzare

Sistemi Embedded
2010/2011

8

Introduzione

- Classe del processore
 - Una prima distinzione riguarda la specializzazione o generalità
 - Nel primo caso si parla di *Application Specific Processor*
 - Microprocessori studiati per applicazioni specifiche quali, per esempio, l'elaborazione numerica dei segnali, l'elaborazione di immagini o la gestione dei pacchetti in applicazioni di rete
 - Come alternativa si hanno i *General Purpose Processor*
 - Microprocessori per applicazioni generiche che non è pensata né ottimizzata per alcuna applicazione in particolare ma, al contrario, si presta bene alla soluzione di problemi di natura diversa
 - Nei sistemi embedded multiprocessore una soluzione tipica consiste nell'uso di un GPP per la supervisione e la gestione delle attività e di ASP per le specifiche funzioni

Sistemi Embedded
2010/2011

9

Introduzione

- Forma in cui acquisire il processore
 - COTS vs IP
 - Nel primo si tratta di comperare fisicamente il chip, montarlo su una board e di interfacciarlo a tutti i restanti dispositivi del sistema
 - Nel secondo caso, invece, si acquista una descrizione del processore a un livello di astrazione differente a seconda dei casi
 - Si spazia dalle *soft macro*, in cui il processore è descritto in HDL a livello RT fino alle *hard macro* in cui è invece descritto a livello di layout
 - Data la grande disponibilità di risorse logiche sui PLD, l'uso di IP sta diffondendosi rapidamente e ogni produttore di CPLD e/o FPGA fornisce uno o più core
 - » Micro/Pico Blaze di Xilinx, Nios-II di Altera, Mico32 di Lattice, ARM7 di Arm Ltd. e Actel

Sistemi Embedded
2010/2011

10

Introduzione

- Classe & Forma
 - La scelta della classe di processore da utilizzare è quasi esclusivamente legata alla natura del problema, ovvero, alle caratteristiche dell'algoritmo, mentre la scelta della forma in cui acquisire il processore è prevalentemente legata all'architettura target del sistema
 - SoC, MPSoC, NoC, PCB e così via
 - Questi appena discussi sono i principali fattori che guidano le decisioni del progettista, ma vi sono molti altri aspetti che devono essere presi in considerazione

Introduzione

- Prestazioni
 - Il criterio principale per la valutazione delle prestazioni è una misura del numero medio di istruzioni per ciclo di clock (IPC/CPI)
 - Questa misura è relativa alla frequenza di clock e pertanto deve essere opportunamente scalata per confrontare architetture diverse
 - Una misura assoluta del throughput sono i MIPS ma vanno utilizzati con attenzione poiché i processori hanno istruzioni macchina differenti
 - Una misura analoga è il MFLOPS
 - Per architetture più specifiche, quali i DSP o i NP le misure per la valutazione delle prestazioni possono cambiare
 - Per i DSP si usano spesso i MMACS
 - Per i NP si usa il numero di pacchetti processati per unità di tempo

Introduzione

- Prestazioni
 - Oltre alle classiche misure di prestazioni, è necessario considerare altre grandezze non funzionali tra cui forse la più importante per i sistemi embedded è la dissipazione di potenza
 - Si usano misure quali la potenza media o la potenza di picco
 - Misure sono utili per una valutazione di massima della potenza del sistema complessivo
 - Molto spesso si ricorre a misure che combinano potenza e prestazioni, per esempio i mW/MHz o, a un livello ancora più astratto, i MIPS/mW
 - Anche queste misure, sebbene più utili e ricche d'informazione, devono comunque essere utilizzate con cautela

Sistemi Embedded
2010/2011

13

Introduzione

- Memoria
 - I requisiti di memoria di un'applicazione sono spesso un fattore determinante nella scelta del processore da utilizzare
 - Capacità di memoria e velocità di accesso (o banda)
 - In alcuni sistemi particolarmente critici potrebbe non essere possibile ricorrere a memorie esterne al processore
 - La stessa scelta potrebbe essere dettata dalla necessità di disporre di una banda molto elevata e raggiungibile solo da memorie integrate
 - Per applicazioni e sistemi più complessi, il punto critico è spesso legato alla capacità d'indirizzamento del microprocessore specifico
 - » Architetture semplici, infatti, limitano lo spazio d'indirizzamento a 16 bit (64 KB), 20 bit (1 MB) o 24 bit (16 MB) mentre processori più complessi arrivano fino a 32 bit (4 GB)

Sistemi Embedded
2010/2011

14

Introduzione

- Periferiche

- In un sistema embedded il microprocessore spesso elabora segnali provenienti dall'ambiente e controlla apparati esterni
- E' possibile quindi scegliere tra un'architettura di elaborazione pura e una che integra su single-chip sia la parte di elaborazione sia un insieme di periferiche e interfacce
 - Microprocessori vs. Microcontrollori
 - » La scelta di un microprocessore richiederà l'integrazione su PCB o SoC di tutte le periferiche necessarie a opera del progettista
 - » La scelta di un microcontrollore solleva il progettista dal problema dell'integrazione, ma vincola le possibilità a un insieme ristretto di alternative

Introduzione

- Software

- La disponibilità di componenti software per uno specifico processore è un altro fattore importante da considerare
- Analizzando infatti un ampio spettro di applicazioni embedded reali si scopre che il loro contenuto proprietario (*legacy*) è abbastanza ridotto e gran parte delle funzioni realizzate è invece standard
 - Es. Funzioni di elaborazione numerica (filtri, trasformate di Fourier, trasformate wavelet), funzioni di image processing (filtri convoluzionali, motion detection, trasformate discrete ai coseni)
- La disponibilità di tali librerie semplifica il processo di progettazione e sviluppo di un'applicazione, in alcuni casi al punto da rendere fattibile e conveniente una soluzione SW altrimenti impraticabile
 - Oltre alle librerie, la disponibilità di un sistema operativo per un certo processore è un criterio spesso decisivo nella scelta

Introduzione

- Ambiente di sviluppo
 - Sebbene sia ormai riconosciuto che l'ambiente di sviluppo fornito a supporto di un microprocessore costituisca un fattore determinante per la valutazione della qualità di una soluzione, esistono ancora notevoli differenze tra i vari ambienti SDK
 - Sono due gli aspetti da considerare nella valutazione di tali ambienti: il flusso di compilazione da un lato e l'insieme degli strumenti di analisi dall'altro
 - Per i primi, a essere determinante è la qualità del codice generato e la flessibilità offerta all'utente in termini di controllabilità del flusso
 - Per gli strumenti di analisi, invece, è importante valutare la ricchezza e la varietà, nonché l'accuratezza e l'affidabilità offerte

Introduzione

- Packaging
 - Se il processore viene acquisito sotto forma di componente discreto il tipo di packaging può diventare un aspetto importante
 - Dimensione, piedinatura, materiale
- Certificazione
 - Esistono diversi livelli di certificazione
 - Commerciale, industriale, militare e aerospaziale
 - La disponibilità di un componente con l'adeguata certificazione può essere un fattore vincolante per la sua adozione

Microprocessori

Microprocessori General Purpose

GPP

- Un GPP è un'architettura di calcolo generica, pensata per essere adatta in campi anche molto diversi
 - Utilizzati soprattutto nei PC/PDA
 - Nei sistemi embedded si usano se i requisiti di prestazioni non sono particolarmente spinti e la natura dell'applicazione è eterogenea
 - Operazioni di controllo e supervisione e interfacciamento lento o interattivo anche tramite GUI
- Tra i GPP è possibile distinguere diverse architetture
 - Una prima importante classificazione li distingue in CISC e RISC
 - CISC: numero molto elevato di istruzioni complesse
 - RISC: poche istruzioni semplici

GPP

Architetture CISC

GPP

- Architetture CISC
 - I primi GPP furono strutturati secondo una “filosofia” nota come *Complex Instruction Set Computer*
 - La ragione sta nella disponibilità limitata di risorse di memorizzazione dell'epoca
 - Era necessario studiare delle architetture che permettessero di ottenere, a parità di funzioni svolte, programmi di dimensioni ridotte
 - » Questo ha portato alla costruzione di instruction set e microprocessori in grado di svolgere mediante una sola istruzione le tre operazioni fondamentali: *load*, *execute* e *store*
 - » Inoltre, sempre per fornire funzionalità complesse con singole istruzioni, gli instruction set si sono arricchiti di varie modalità d'indirizzamento

GPP

- Architetture CISC

- Modalità di indirizzamento

Modalità	Esempio	Significato
Assoluto (istruzioni)	<code>jmp a</code>	$PC \leftarrow a$
Relativo (istruzioni)	<code>jmp (a)</code>	$PC \leftarrow PC + a$
Indiretto (istruzioni)	<code>jmp Rs</code>	$PC \leftarrow PC + Rs$
Registro	<code>add Rd, R1, R2</code>	$Rd \leftarrow R1 + R2$
Assoluto	<code>load Rd, a</code>	$Rd \leftarrow M(a)$
Indiretto	<code>load Rd, [Rs]</code>	$Rd \leftarrow M(Rs)$
Base, offset	<code>load Rd, a [Rs]</code>	$Rd \leftarrow M(Rs + a)$
Indice	<code>load Rd, [Rs, Ri]</code>	$Rd \leftarrow M(Rs + Ri)$
Indice, offset	<code>load Rd, a [Rs, Ri]</code>	$Rd \leftarrow M(a + Rs + Ri)$
Indice, offset, scala	<code>load Rd, a [Rs, Ri], s</code>	$Rd \leftarrow M(s \times (a + Rs + Ri))$
Autoincremento	<code>load Rd, + [Rs]</code>	$Rd \leftarrow M(Rs), Rs \leftarrow Rs + 1$
Autodecremento	<code>load Rd, - [Rs]</code>	$Rd \leftarrow M(Rs), Rs \leftarrow Rs - 1$
Push	<code>push Rs</code>	$M(SP) \leftarrow Rs, SP \leftarrow SP + 1$
Push	<code>pop Rd</code>	$SP \leftarrow SP - 1, Rd \leftarrow M(SP)$

GPP

- Architetture CISC

- Idealmente ogni istruzione aritmetico/logica dovrebbe poter accedere ai dati e scrivere il risultato tramite una qualsiasi delle modalità d'indirizzamento disponibili
 - Architettura o Instruction Set perfettamente *ortogonale*
- I casi reali tendono a questa situazione, ma limitano il numero di combinazioni possibili delle varie modalità d'indirizzamento
 - Normalmente, uno degli operandi si riferisce alla memoria, mentre l'altro è un registro e funge sia da sorgente/destinazione
 - Nonostante questa semplificazione, le istruzioni CISC sono di norma codificate da parole di lunghezza variabile da caso a caso
 - » Questa circostanza non solo complica notevolmente le unità di fetch e decode, ma aumenta i requisiti di velocità di accesso alla memoria

GPP

- Architetture CISC

- Per quanto riguarda le funzioni svolte dalla ALU, si ha in genere una grande ricchezza di varianti delle operazioni di base
 - Somma/sottrazione, moltiplicazione, divisione, operazioni logiche, di shift e di confronto
 - Nelle architetture più moderne si hanno poi alcune istruzioni cosiddette vettoriali, che cioè lavorano su più registri contemporaneamente
 - » *Single Instruction Multiple Data (MMX,..., SSE, AVX)*
- Una così ampia varietà di modalità d'indirizzamento e di operazioni aritmetico logiche richiede un *data-path* molto complesso e, di conseguenza, meno veloce

Sistemi Embedded
2010/2011

25

GPP

- Architetture CISC

- Varianti dell'istruzione ADD

Istruzione	Significato
add	Istruzione di base
adc	Somma con riporto in ingresso
addi	Somma con una costante
addi	Somma con una costante a 32 bit
addq	Somma con una costante a 3 bit
aaa	Somma di operandi unpacked BCD
daa	Somma di operandi packed BCD
xadd	Somma e scambia gli operandi
addps	Somma di operandi a 128 bit
addss	Somma dei 32 LSB di operandi a 128 bit
paddb	Somma i singoli byte
paddb	Somma i singoli byte con saturazione
adawl	Somma operandi allineati alla parola
addb3	Somma tre operandi a 8 bit
addw3	Somma tre operandi a 32 bit
addl3	Somma tre operandi a 64 bit

Sistemi Embedded
2010/2011

26

GPP

- Architetture CISC
 - Per aumentare la velocità di esecuzione in termini di *throughput* i processori CISC dispongono di pipeline molto complesse (fino a 23 stadi) e di difficile gestione
 - Una ulteriore strategia, anch'essa costosissima in termini di risorse hw necessarie, si basa sul fatto che in molti casi l'ordine di esecuzione delle istruzioni assembly di un programma può essere alterato senza modificarne la semantica
 - Con una pipeline così complessa è molto difficile per un **compilatore** prevederne lo stato e la disponibilità al momento dello scheduling delle istruzioni per cui questo aspetto spesso viene ignorato
 - » Le istruzioni arrivano quindi alla decodifica in un ordine potenzialmente non ottimale rispetto allo stato istantaneo della pipeline

GPP

- Architetture CISC
 - Affinché una strategia simile possa risultare vantaggiosa è necessario conoscere lo stato della pipeline
 - Le moderne architetture CISC dispongono di specifiche unità di controllo per effettuare un nuovo scheduling dinamicamente
 - Out-of-order execution
 - La realizzazione di una tale logica è molto complessa e comporta un incremento della dimensione del processore
 - Come conseguenza, benché le prestazioni di un CISC siano del tutto ragguardevoli, se considerate in relazione alla potenza assorbita cadono di molto al di sotto dei livelli di efficienza raggiunti da altre tipologie di architetture

GPP

Architetture RISC

GPP

- Architetture RISC
 - Si dice *Restricted Instruction Set Computer* un'architettura con poche semplici istruzioni
 - Alla fine degli '80 le DRAM hanno ridotto il costo per Kb
 - Ciò ha reso meno critico il problema della dimensione dei programmi
 - » Le operazioni CISC scomposte in istruzioni RISC semplificano l'architettura
 - Nello stesso periodo si ha una crescita della capacità d'integrazione
 - Realizzazione su singolo chip di funzioni e architetture complesse
 - » Pipeline e numero elevato di registri general purpose
 - » Per sfruttare la pipeline il tempo di esecuzione delle istruzioni deve essere uniforme
 - » Questo è più facilmente realizzabile se le istruzioni sono semplici

GPP

- Architetture RISC

- Il principale vantaggio di un processore RISC sta nella semplicità di istruzioni e architettura
 - La decodifica di una istruzione semplice è meno complessa
 - Le istruzioni RISC sono spesso di dimensione fissa
 - » Ciò permette di bilanciare gli stadi e aumentare la frequenza di clock
- Anche l'esecuzione (ALU e BPU) beneficia della semplicità
 - Le architetture RISC operano esclusivamente su registri e pertanto sono più veloci e caratterizzate da una durata predicibile
 - I processori RISC dispongono di un elevato numero di registri
 - » Tale numero di registri consente di ridurre significativamente il ricorso al *memory spill*

GPP

- Architetture RISC

- L'eccezione sono le operazioni di I/O, ovvero le operazioni di caricamento dei dati dalla memoria nei registri e viceversa
 - Ogni istruzione di calcolo necessita di un preventivo caricamento e, eventualmente, di una scrittura dei risultati in memoria
 - **Architetture load/store**
- Le operazioni di trasferimento dei dati hanno quindi una notevole importanza e sono pertanto un aspetto critico
 - Una primo accorgimento sta nel fatto che tali istruzioni dispongono di poche modalità d'indirizzamento piuttosto elementari
 - Inoltre, è necessario ridurre il tempo effettivo di accesso ai dati
 - » La soluzione a questo problema è offerta dalla strutturazione di una opportuna gerarchia di memoria basata sull'uso di cache

GPP

- Architetture RISC
 - Modalità di indirizzamento

Modalità	Esempio	Significato
Relativo (istruzioni)	<code>jmp (a)</code>	$PC \leftarrow PC + a$
Indiretto (istruzioni)	<code>jmp Rs</code>	$PC \leftarrow PC + Rs$
Registro	<code>add Rd, R1, R2</code>	$Rd \leftarrow R1 + R2$
Base, offset	<code>load Rd, a[Rs]</code>	$Rd \leftarrow M(Rs + a)$
Autoincremento	<code>load Rd, +[Rs]</code>	$Rd \leftarrow M(Rs), Rs \leftarrow Rs + 1$
Autodecremento	<code>load Rd, -[Rs]</code>	$Rd \leftarrow M(Rs), Rs \leftarrow Rs - 1$
Multiplo	<code>load {R1, R2, ...}, [Rs]</code>	$R1 \leftarrow M(Rs), R2 \leftarrow M(Rs+1), \dots$

GPP

- Architetture RISC
 - Un aspetto importante riguarda il flusso SW
 - A parità di codice sorgente il numero di istruzioni prodotte dal compilatore sarà più elevato rispetto ai CISC
 - Le architetture RISC non dispongono di unità HW dedicate al rilevamento e alla gestione dei conflitti della pipeline
 - » Ma la predicibilità del tempo di esecuzione delle istruzioni, permette a un buon compilatore di riorganizzare la struttura del programma
 - Un'ultima considerazione riguarda la potenza
 - I RISC sono caratterizzati da un'architettura semplice e presentano un consumo di potenza molto basso
 - 30-100 mW/MHz o intorno al mW/MIPS

GPP

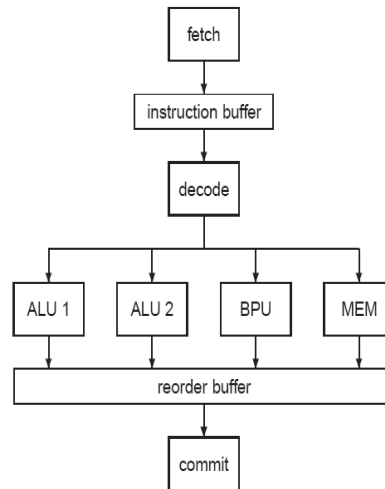
Architetture Superscalari

GPP

- Architetture Superscalari
 - L'analisi di molti programmi mostra che la natura sequenziale dell'assembly è restrittiva, poiché la semantica del programma evidenzia il cosiddetto parallelismo a livello d'istruzione
 - Esempio: istruzioni $y=a+b$ e $z=c*d$
 - Non esiste alcuna dipendenza tra le due e pertanto possono essere eseguite in qualsiasi ordine e, al limite, contemporaneamente
 - Questa considerazione è la motivazione principale che sta alla base dello sviluppo e dell'efficienza delle architetture superscalari
 - Si dice superscalare un'architettura dotata di più unità di elaborazione, cioè di più ALU e/o BPU

GPP

- Architetture Superscalari
 - Tipica architettura superscalare



Sistemi Embedded
2010/2011

37

GPP

- Architetture Superscalari
 - Un tale schema, grazie al parallelismo offerto dallo stadio execute, permette di eseguire più istruzioni per ciclo di clock
 - Inevitabilmente la complessità della logica di controllo cresce a discapito della frequenza di clock e della potenza dissipata
 - Inoltre, si tenga presente che è il microprocessore a definire dinamicamente lo scheduling delle istruzioni
 - Pro
 - Il compilatore non deve occuparsi di riorganizzare il codice esplicitamente per una specifica struttura hardware
 - Contro
 - Una sostanziale porzione del processore è dedicata a implementare i complessi meccanismi di scheduling e a gestire la consistenza

Sistemi Embedded
2010/2011

38

GPP

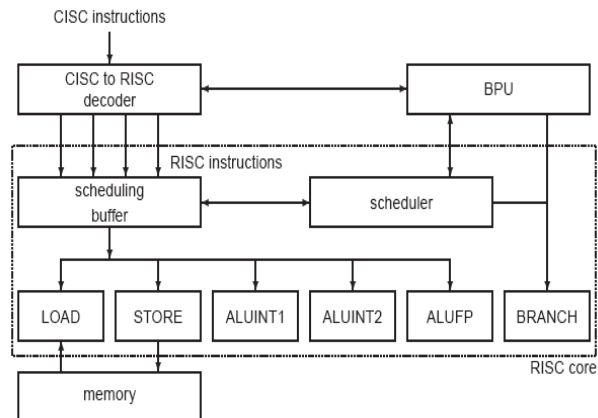
Architetture CISC/RISC

GPP

- Architetture CISC/RISC
 - Approccio che combina *out-of-order execution* e superscalarità è proposto con l'obiettivo di mantenere la compatibilità a livello di istruzioni assembly con le versioni precedenti di processori
 - L'Instruction set x86 è di fatto lo standard CISC a livello mondiale
 - L'approccio consiste nello scomporre ogni istruzione CISC in una sequenza di istruzioni elementari e farle eseguire da un core semplice ed efficiente
 - Benché estremamente onerosa in termini di risorse HW, questa soluzione unisce i vantaggi della retrocompatibilità con le elevate prestazioni raggiungibili da un core di elaborazione RISC

GPP

- Architetture CISC/RISC
 - Struttura di un processore CISC con core RISC



Sistemi Embedded
2010/2011

41

GPP

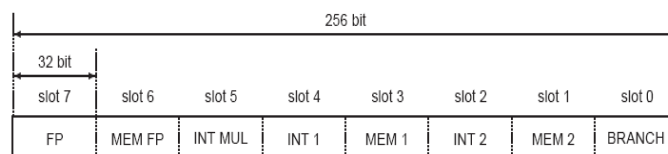
Architetture EPIC/VLIW

GPP

- Architetture EPIC/VLIW
 - Per superare il limite legato alla complessità dello scheduling HW delle istruzioni, Intel per prima ha sviluppato una nuova classe di processori noti come EPIC (o VLIW) di cui l'*Itanium* è stato il primo esempio commerciale
 - Queste architetture supportano il parallelismo esplicito
 - Capacità di eseguire più istruzioni simultaneamente e sotto il controllo del programma
 - Una istruzione VLIW è costruita impaccando più istruzioni elementari, tipicamente RISC, in una sola parola di grandi dimensioni avente in genere una struttura ben definita e fissa

GPP

- Architetture EPIC/VLIW
 - Struttura di una istruzione VLIW



GPP

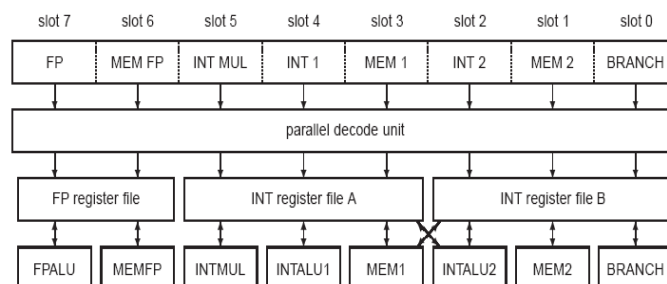
- Architetture EPIC/VIW
 - Tale parola, prelevata dalla memoria nella fase di fetch, viene decodificata in parallelo
 - Al termine della decodifica le singole istruzioni RISC corrispondenti ai vari slot vengono inviate alle corrispondenti unità di esecuzione secondo uno schema fissato
 - Data una tale struttura rigida, è il compilatore a dover effettuare un lavoro di scheduling complesso e articolato
 - Gli algoritmi di ottimizzazione, di conseguenza, devono essere esplicitamente strutturati per una specifica architettura

Sistemi Embedded
2010/2011

45

GPP

- Architetture EPIC/VIW
 - Per consentire l'esecuzione parallela di più istruzioni, i register file, cioè i banchi di registri dati, di tali processori devono essere dotati di più porte di lettura e devono essere connessi alle varie unità in modo molto flessibile



Sistemi Embedded
2010/2011

46

GPP

- Architetture EPIC/VLW
 - L'approccio EPIC/VLW ha quindi come idea di base lo spostamento di tutta la complessità dello scheduling e dell'ottimizzazione sul compilatore
 - Ne risultano architetture semplici, benché spesso di notevoli dimensioni a causa del numero elevato di unità di esecuzione
 - Il consumo di potenza si attesta a un livello intermedio tra le architetture RISC semplici e quelle superscalari *pure*
 - Simili soluzioni, permettendo di raggiungere prestazioni molto elevate, sono state adottate anche da microprocessori non general purpose, quali per esempio i *digital signal processor*

Sistemi Embedded
2010/2011

47

GPP

Architetture CISC/VLW

GPP

- Architetture CISC/VLIW
 - Approccio analogo al CISC/RISC che combina le prestazioni VLIW con l'obiettivo di mantenere la compatibilità a livello di istruzioni assembly con le versioni precedenti di processori
 - L'approccio consiste nello scomporre ogni istruzione CISC in istruzioni elementari VLIW e farle eseguire da un core semplice ed efficiente
 - Esempio famoso (unico?)
 - Transmeta Crusoe
 - » Oggi disponibile solo come IP core

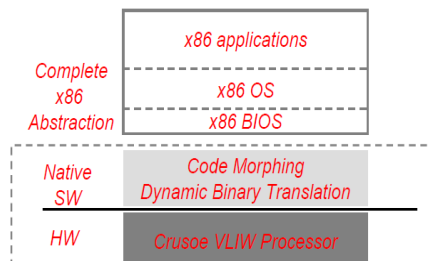
Sistemi Embedded
2010/2011

49

GPP

- Architetture CISC/VLIW
 - Esempio famoso

Transmeta Crusoe & Code Morphing



- Crusoe boots "Code Morpher" from ROM at power-up
- Crusoe+Code Morphing == x86 processor
x86 software (including BIOS) cannot tell the difference

Sistemi Embedded
2010/2011

50

GPP

Analisi comparata

	RISC	CISC	Superscalari	EPIC/VLIW
Instruction set	semplice	complesso	complesso	semplice
Modi d'indirizzamento	pochi	molti	molti	pochi
Dimensione istruzioni	fissa	variabile	variabile	fissa, grande
Register file	singolo ⁽¹⁾	singolo ⁽¹⁾	singolo ⁽¹⁾	multiplo
Numero registri	alto	basso	medio	molto alto
Scheduling istruzioni	statico	dinamico	dinamico	statico
Gestione conflitti	statico ⁽²⁾	dinamico	dinamico	statico
Complessità compilatore	media	alta	bassa	molto alta
Parallelismo	assente	assente	implicito	esplicito
Complessità hardware	bassa	alta	molto alta	alta
Complessità pipeline	bassa	alta	molto alta	media ⁽³⁾
Consumo di potenza	molto basso	alto	molto alto	alto
IPC tipico	≈ 1	< 1	> 1	$\gg 1$

⁽¹⁾In alcuni casi disgiunto tra registri interi e registri floating-point.

⁽²⁾I conflitti vengono riconosciuti e risolti unicamente mediante stalli.

⁽³⁾Struttura semplice, ma di grandi dimensioni per via delle molte unità di elaborazione.

Sistemi Embedded
2010/2011

51

Microprocessori

Microprocessori Application Specific

ASP

- Nel mondo embedded ci si trova a risolvere problemi estremamente precisi e specifici e a sviluppare sistemi in grado di svolgere prevalentemente una o poche funzioni
 - Data la natura molto eterogenea dei problemi che si incontrano, non sempre una soluzione basata su GPP risulta soddisfacente
- Quindi, per diversi settori applicativi, sono state progettate architetture ottimizzate
 - Tali microprocessori dedicati si differenziano in termini di instruction set, architettura, dimensioni, potenza di calcolo e dissipazione di potenza
 - Digital Signal Processor, Network Processor, Microcontrollori
 - Di recente: DSC, mixed (e.g. PSoC della Cypress o DSP+FPAA)

Sistemi Embedded
2010/2011

53

ASP

Digital Signal Processor

ASP

- Digital Signal Processor

- I digital signal processor o DSP sono per eccellenza i processori dedicati più noti e di maggiore utilizzo
 - Indipendentemente dalle moltissime, e in alcuni casi sostanziali, differenze tra i vari dispositivi, tutti i DSP sono accomunati dal fatto di essere strutturati in modo specifico per l'elaborazione numerica
- Per comprendere le ragioni alla base delle scelte architetture adottate nei DSP è importante analizzare la struttura tipica degli algoritmi di elaborazione numerica

ASP

- Digital Signal Processor

- Tipiche operazioni degli algoritmi di signal processing

Operazione	Forma chiusa	Forma iterativa
Somma vettoriale	$\mathbf{Z} = \mathbf{X} + \mathbf{Y}$	$z_n = x_n + y_n \quad \forall n$
Riscaldamento	$\mathbf{Z} = k\mathbf{X}$	$z_n = k \cdot x_n \quad \forall n$
Somma	$z = \mathbf{X} _1$	$z = \sum_n x_n$
Somma quadratica	$z = \mathbf{X} _2$	$z = \sum_n x_n^2$
Prodotto scalare	$z = \mathbf{X} \times \mathbf{Y}$	$z = \sum_n x_n \cdot y_n$
Convoluzione	$\mathbf{Z} = \mathbf{X} * \mathbf{Y}$	$z_n = \sum_k x_k \cdot y_{N-k} \quad \forall n$
Trasformata di Fourier	$\mathbf{Z} = \mathcal{F}(\mathbf{X})$	$z_n = \sum_k x_k e^{-\frac{2\pi i}{N}nk} \quad \forall n$

ASP

- Digital Signal Processor
 - Operazione predominante

$$y_i = \sum_{k=0}^{n-1} x_{i-k} * a_k$$

$$y_{i,j} = y_{i,j-1} + x_{i-j} * a_j$$

$$z_{t+1} = z_t + x * y$$

- Questa operazione è talmente importante nelle applicazioni DSP che le è stato assegnato un nome specifico
 - **MAC : Multiply Accumulate**
- Questa prima considerazione porta alla conclusione che una buona architettura per DSP deve ottimizzare il più possibile le operazioni di somma e moltiplicazione, eventualmente combinandole in un unico operatore a tre ingressi

ASP

- Digital Signal Processor
 - Inoltre, per la natura delle operazioni analizzate, una parte consistente di un'algoritmo numerico sarà costituita da cicli
 - Se in generale non è semplice ottimizzare un'architettura programmabile in modo da rendere molto efficiente la gestione dei cicli, nel caso dei DSP questo è fattibile per le seguenti ragioni
 - Il corpo del ciclo è spesso di piccole dimensioni (~10 istruzioni asm)
 - La variabile di controllo del ciclo non viene alterata nel corpo del ciclo
 - La variabile di controllo del ciclo viene aggiornata in modo semplice
 - Il pattern di accesso ai dati dei vettori coinvolti nel corpo del ciclo è generalmente semplice e regolare

ASP

- Digital Signal Processor
 - Queste caratteristiche rendono possibile la realizzazione di architetture HW per l'ottimizzazione dei cicli
 - Buffer circolare destinato a contenere le istruzioni del corpo del ciclo
 - Registro dedicato alla gestione della variabile di controllo
 - Sommatore/sottrattore per svolgere le operazioni d'incremento/decremento senza utilizzare la ALU
 - Perché una tale struttura possa portare un beneficio l'accesso ai dati in memoria deve essere sia estremamente rapido
 - Apposite architetture che offrono prestazioni elevate

ASP

- Digital Signal Processor
 - Gerarchie di memoria per DSP
 - Bus ad alta velocità e con una larghezza di parola fino a 256 bit
 - Classica struttura gerarchica con vari livelli di cache
 - Architettura classica
 - » Cache unificata per dati e istruzioni
 - Architetture dedicate
 - » Harvard: cache L0 separata per dati e istruzioni
 - » SHARC: tre cache L0 per dati, istruzioni e costanti

ASP

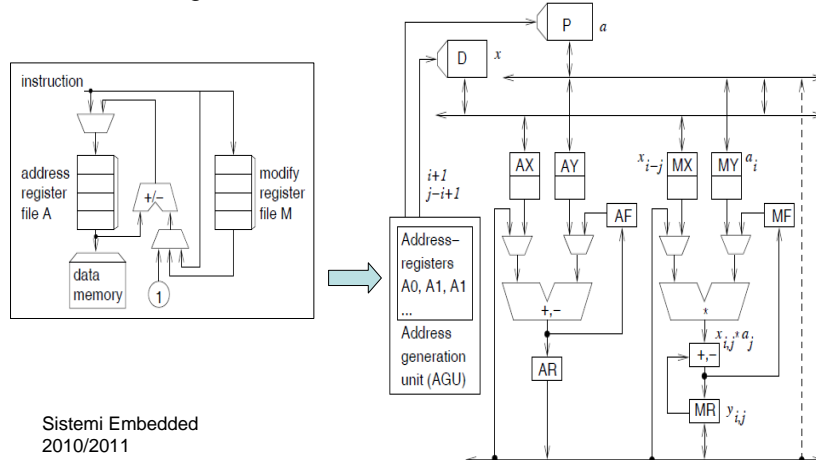
- Digital Signal Processor

- Oltre alla notevole flessibilità e potenza di calcolo offerta da tali istruzioni e dalle speciali ALU che le realizzano, i moderni DSP sono spesso strutturati secondo l'approccio VLIW
 - Per la natura specifica delle applicazioni di calcolo numerico, risulta particolarmente adatto
- Tali applicazioni sono infatti caratterizzate da una scarsissima parte di controllo e un elevatissimo parallelismo intrinseco

ASP

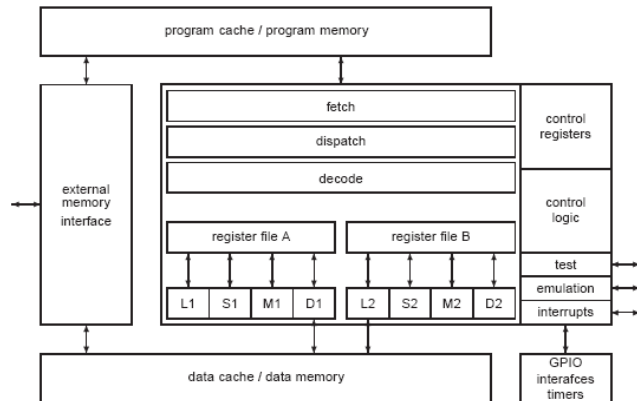
- Digital Signal Processor

- Analog Devices ADSP 2100



ASP

- Digital Signal Processor
 - Architettura di un DSP VLIW della TEXAS



Sistemi Embedded
2010/2011

63

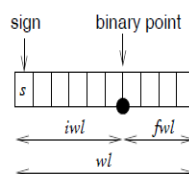
ASP

- Digital Signal Processor
 - Altre particolarità

- Saturating Arithmetic

	0111
+	1001
Standard wrap-around arithmetic	10000
saturating arithmetic	1111

- Fixed-point arithmetic



Sistemi Embedded
2010/2011

64

ASP

Network Processor

ASP

- Network Processor
 - Sono architetture dedicate alla elaborazione di pacchetti
 - Sono tipicamente strutturati come complessi system-on-chip con un elevato grado di parallelismo
 - I NP, sono architetture dedicate alle applicazioni di rete
 - Il mondo delle reti è estremamente ampio e si arricchisce rapidamente di nuovi protocolli e nuovi standard per cui definire in modo generale le tipiche operazioni di un network processor è tutt'altro che semplice

ASP

- Network Processor
 - Senza entrare nello specifico, tuttavia, alcune classi di operazioni possono essere individuate
 - Buffering dei pacchetti in ingresso/uscita
 - Rimozione e aggiunta degli header a livello data-link
 - Ricerca di uno specifico campo nell'header IP
 - Ricerca di un indirizzo in una lookup table
 - Calcolo di codici a ridondanza ciclica
 - Invio o eliminazione dei pacchetti

ASP

- Network Processor
 - Per realizzare tali funzioni garantendo le elevate prestazioni richieste dalle moderne reti, i NP si appoggiano a diverse strutture hardware appositamente ottimizzate allo scopo
 - Interfacce di I/O e memorie molto veloci, code, unità hardware dedicate, core crittografici e uno o più core di microprocessori RISC
 - Molto spesso le applicazioni di *packet processing* richiedono la gestione di più canali indipendenti che chiaramente beneficiano di una elaborazione parallela
 - Ogni canale è gestito da una unità hardware dedicata, spesso indicata come PP (*Packet Processor*) o CP (*Channel Processor*)

ASP

- Network Processor

- In alcuni casi, tuttavia, esistono relazioni tra i flussi separati che devono essere prese in considerazione
 - Si ricorre in questo caso core RISC che, opportunamente programmati, fungono da supervisori e realizzano tutte le operazioni di controllo e gestione di più alto livello
- Una delle caratteristiche del *packet processing* è che una tipica applicazione è strutturata come un insieme di semplici routine, ognuna delle quali lavora su una grande quantità di pacchetti
 - In altre parole, quindi, il packet processing è caratterizzato da programmi piccoli che lavorano su enormi quantità di dati

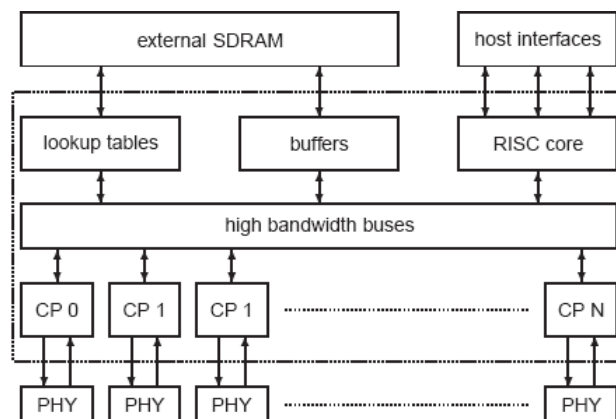
Sistemi Embedded
2010/2011

69

ASP

- Network Processor

- Architettura tipica di un NP



Sistemi Embedded
2010/2011

70

ASP

- Network Processor
 - Una volta compresa l'architettura di massima di questo tipo di processori, è utile discutere della loro programmazione
 - Core RISC
 - Sviluppo di programmi, tipicamente a partire da codice sorgente in C, per un normale RISC e, in quanto tale, non presenta alcuna peculiarità se non la funzione svolta
 - Singoli PP
 - La programmazione dei PP avviene invece spesso a livello assembly ed è caratterizzata dalla sua struttura fortemente parcellizzata in routine semplici
 - » L'approccio alla programmazione dei CP è pertanto scarsamente strutturato e viene prevalentemente svolto in modo manuale tranne in rari casi in cui si può seguire un approccio funzionale basato sul *pattern recognition*

Sistemi Embedded
2010/2011

71

ASP

Microcontrollori

ASP

- Microcontrollori

- Con il termine microcontrollore (MCU – *Micro Controller Unit*) si indica una classe di microprocessori che dispongono di molte periferiche e interfacce integrate su single-chip
 - I MCU più complessi possono avere anche uno o più *single-purpose processor*
- L'aspetto importante sta proprio nella loro natura di SoC
 - Questa caratteristica li rende particolarmente adatti per risolvere problemi che richiedono un carico computazionale modesto, una struttura dell'applicazione decisamente semplice, ma che pongono molti vincoli stringenti sull'uso di risorse hardware
 - » Una soluzione integrata quindi risulta sicuramente adatta ed efficiente

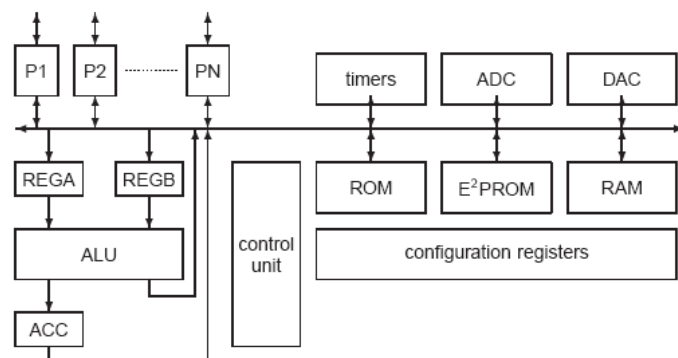
Sistemi Embedded
2010/2011

73

ASP

- Microcontrollori

- Architettura tipica di un semplice MCU



Sistemi Embedded
2010/2011

74

ASP

- Microcontrollori

- Nello schema mostrato, si nota l'assenza di una interfaccia verso una memoria esterna e questa è una caratteristica comune
 - I programmi per MCU sono in genere piccoli
 - Una interfaccia di memoria aumenterebbe il numero di pin
- In sistemi fortemente vincolati l'ingombro geometrico di un componente può essere un fattore cruciale
 - Per fornire una interfaccia di memoria senza aumentare il numero di pin, le architetture prevedono un uso multiplexato delle porte di I/O
 - Secondo questo schema, la logica di controllo delle porte deve essere configurata in modo da selezionarne la modalità

Sistemi Embedded
2010/2011

75

ASP

- Microcontrollori

- Grazie ai registri di configurazione e limitando il numero di pin, le architetture offrono diverse periferiche e interfacce
 - I2C, SPI, CAN, JTAG, modulatori *PWM*, *UART/USART*, *watchdog*, *timer*, ingressi e comparatori analogici, ecc...
- Lo sviluppo del software per MCU avviene tipicamente in C, anche se in molti casi si ricorre ancora all'assembly per ottenere migliori prestazioni e compattezza
- Gli ambienti di sviluppo per microcontrollori variano in complessità da semplici compilatori C, assembler e debugger fino ad ambienti integrati che offrono un supporto completo

Sistemi Embedded
2010/2011

76

ASP

Multi-Processor SoC

ASP

- Multi-Processor SoC
 - Further increase of clock rates of processors has recently come to a standstill: the large energy consumption of processors using multi-gigahertz clock speeds is a key reason for this
 - In order to still improve the overall processing power, several processors must be employed
 - This led to the design of chips comprising multiple processors as well as additional components such as peripheral devices and memories
 - Systems implemented in that way are called **MPSoC**

ASP

- Multi-Processor SoC

- For general purpose computing and PCs, multi-processor systems are typically homogeneous (all processors are of the same type)
 - Multi-core
- For embedded systems, energy efficiency has top priority and it is typically obtained with highly specialized processors
 - In order to save energy, unused areas are typically powered-down
- However, using such multiprocessor-based systems from applications written in a sequential language is a challenge
 - Mapping techniques for such processors are important, since examples demonstrate that a power efficiency close to that of ASIC can be achieved

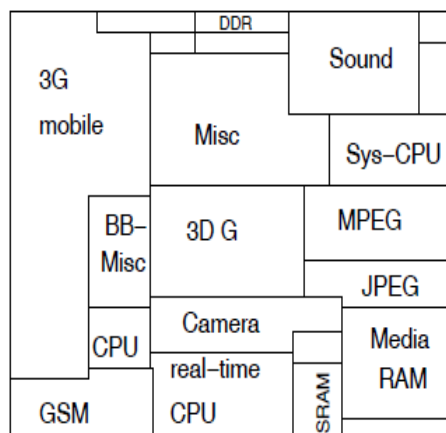
Sistemi Embedded
2010/2011

79

ASP

- Multi-Processor SoC

- Renesas SH-Mobile G1



Sistemi Embedded
2010/2011

80

Conclusioni

Conclusioni

- L'argomento discusso in questa parte è molto vasto e una trattazione non approfondita come quella svolta qui non può che essere incompleta
 - L'ottica fornita, tuttavia, non è verticale e di dettaglio, ma piuttosto orizzontale e comparativa, con l'obiettivo di aiutare a comprendere e individuare i criteri di scelta per orientarsi nel vastissimo mondo dei microprocessori