Computer Organization and Assembly Languages Final Project

The comparison between CISC&RISC CPU instruction set

T96901001　曹瀟然

2008/1/20

Introduction

處理器的指令集可分為CISC(complex instruction set)和RISC(reduced instruction set)，在現代的處理器指令集設計中，兩者均有廣泛的應用，本文主要是兩種指令集特性及其應用的比較。

History

傳統的CISC結構有其固有的缺點,即隨著電腦技術的發展而不斷引入新的複雜的指令集,為支援這些新增的指令,電腦的體系結構會越來越複雜,然而,在CISC指令集的各種指令中,其使用頻率卻相差懸殊,大約有 20％的指令會被反復使用,占整個程式碼的80％。而餘下的80％的指令卻不經常使用,在程式設計中只占20％,顯然,這種結構是不太合理的。

基於以上的不合理性,1979年美國加州大學伯克萊分校提出了RISC的概念,RISC並非只是簡單地去減少指令,而是把著眼點放在了如何使電腦的結構更加簡單合理地提高運算速度上。RISC結構優先選取使用頻最高的簡單指令,避免複雜指令;將指令長度固定,指令格式和尋地方式種類減少;以控制邏輯為主,不用或少用微碼控制等措施來達到上述目的。

到目前為止,RISC體系結構也還沒有嚴格的定義,一般認為,RISC體系結構應具有如下特點：

－ 採用固定長度的指令格式,指令歸整、簡單、基本定址方式有2～3種。

－ 使用單週期指令,便於流水線操作執行。

－ 大量使用register,資料處理指令只對register進行操作,只有載入/ 存儲指令可以訪問記憶體,以提高指令的執行效率。

當然,和CISC架構相比較,儘管RISC架構有上述的優點,但決不能認為RISC架構就可以取代CISC架構,事實上,RISC和CISC各有優勢, 而且界限並不那麼明顯。現代的CPU往往採用CISC的週邊,內部加入了RISC的特性,如超長指令集CPU就是融合了RISC和CISC的優勢,成為未來的CPU發展方向之一。

CISC的主要應用為Inter 80X86系列，RISC的主要應用為ARM系列。

Comparison between RISC and CISC

下面以ARM及Inter IA-32分別作為RISC和CISC的例子，對兩者加以比較。

Registers

兩者均有若干32-bit的registers.

ARM在user-mode下，可以使用15個32-bit general purpose registers (R0-R14), program counter(PC) 和 CPSR。

IA-32在protected-mode下，有若干32-bit General-Purpose Registers和16-bit Segment Registers。由於需要較多的微碼控制，所以有EFLAGS等register用來存儲狀態。

Memory Management

兩者的memory management并無本質不同，其地址分布均為$0 \sim 2^{32} - 1$

IA-32中，爲了實現multitasking，在protected mode下，每個program會被分配一定的的memory空間，并且不會被其它program所更改。

Instruction Set

ARM：

除了thumb-mode外，ARM的指令均為32-bit，其指令集包括：

Data processing (Arithmetic and Logical)

所有數據處理指令的結果均為32-bit，并存儲在register中，其形式為1個operand加上3個addresses，主要指令如下：

ADD，ADC，SUB，SBC，RSB，RSC

AND，ORR，EOR，BIC

MOV，MVN

CMP，CMN，TST，TEQ

Data movement

包括3種類型：

Single register load/store：LDR，STR

Multiple register load/store：LDM，STM

Single register swap：SWP(B)

Flow control

BL，BEQ，BNE，BGE，BLT…..

此外，幾乎所有的ARM指令都可以加上條件控制，減少代碼量并且提高速度，例如：

CS：Carry set

EQ：Equal（Zero set）

LT：Less than

HI：Higher than

全部的ARM Instruction set 如下：

Operation Mnemonic	Meaning	Operation Mnemonic	Meaning
ADC	Add with Carry	MVN	Logical NOT
ADD	Add	ORR	Logical OR
AND	Logical AND	RSB	Reverse Subtract
BAL	Unconditional Branch	RSC	Reverse Subtract with Carry
B⟨*cc*⟩	Branch on Condition	SBC	Subtract with Carry
BIC	Bit Clear	SMLAL	Mult Accum Signed Long
BLAL	Unconditional Branch and Link	SMULL	Multiply Signed Long
BL⟨*cc*⟩	Conditional Branch and Link	STM	Store Multiple
CMP	Compare	STR	Store Register (Word)
EOR	Exclusive OR	STRB	Store Register (Byte)
LDM	Load Multiple	SUB	Subtract
LDR	Load Register (Word)	SWI	Software Interrupt
LDRB	Load Register (Byte)	SWP	Swap Word Value
MLA	Multiply Accumulate	SWPB	Swap Byte Value
MOV	Move	TEQ	Test Equivalence
MRS	Load SPSR or CPSR	TST	Test
MSR	Store to SPSR or CPSR	UMLAL	Mult Accum Unsigned Long
MUL	Multiply	UMULL	Multiply Unsigned Long

X86ASM：

相比ARM而言，X86asm的Instruction Set則複雜很多，除了一些上述的基本指令外，還包括很多更高級的指令，例如：

PUSH，POP

BSWAP：交換32-bit register 中bit的順序

XADD：先交換再累加

串指令MOVS，SCAS，LPDS….

以及大量的Conditional jumps指令

而爲了增強CPU的Multi Media及圖像處理等方面的能力，而設計的MMX，SSE等則是更複雜的指令集。

X86asm Instruction set的複雜度是有其必要性的，因為很多程式中都需要執行較為複雜的操作，因此將一些操作設計成單條指令，能夠大量的減少代碼量及程式的設計難度，使得其可以實現更多的功能。但同時增添了多種尋址方式，增長了指令周期，降低了速度。

因此，RISC和CISC的不同主要體現在Instruction set上，前者只包括最常用的指令，而後者則添加了很多複雜指令，提供更多的功能，但同時也降低了速度，這也是CISC的劣勢，下面將對兩者的優劣加以比較分析。

RISC vs CISC

現代的CISC中包含了很多複雜指令，這些指令的實現其實是相當于將其分解為多條更為簡單的指令運行。其優點就是簡化代碼，從而提供更多的功能，但由於指令的執行時間各不相同且差別很大（從幾個時鐘周期到幾十個不等）使得pipeline難以設計且不能完全填滿，這樣就造成了時間的浪費從而降低了速度。

相比而言，RISC則具有以下優點：

1. RISC 指令系統較小：種類的數量較少，只提供簡單指令。這些指令大多都在 4，5 個時鐘週期內完成。
2. 指令的運算元必須預存於 register 中，這樣操作的時間也統一了。
3. 指令長度，定址方式，格式都整齊劃一：這樣可以充分利用流水線，基本上可實現一個時鐘脈衝執行一條指令的目標。
4. RISC 的副程式調用與 CISC 的不同：在 CISC 中，程式調用、返回時需將上下文保存在堆疊中，需要記憶體操作。而 RISC 將它們存放在 register 中，而且參數也使用 register 傳遞。
5. RISC 中斷可視為特殊的副程式鏈結：CISC 中發生中斷時，所有的 register 內容都被壓入堆疊，而 RISC 對中斷進行區分對待，分為羽量級和重量級。

對羽量級中斷 只保存需要保存的 register 內容；對重量級中斷的處理如同常規中斷。

6. RISC 都採用 piplining、快取記憶體、不使用微碼。

因此，RISC 的速度要高過 CISC，以 1984 年 Motorola 的 CISC 體系的 CPU HC11 為例，這種系列的微處理器擁有 300 多條指令，代碼密度分佈非常合理。由於所有的複雜指令都是在核心中完成的，所以執行每條指令的時間就很長。指令週期從 8 個時鐘週期到 164 個時鐘週期。核心過大導致處理器的最高主頻不超過 12MHz，其性能要遠遠低於同時期的 RISC 機型。

同時，RISC 也有它的缺點：代碼密度不高，可執行檔體積較大，彙編代碼可讀性較差。代碼密度不高是個值得關注的問題：若不使用 cache，會需要更大的指令存儲空間，取指時也佔用更大的記憶體帶寬。若採用 cache，又會降低 cache 的命中率。

而從 CPU 的設計上來講，由於 RISC 的核心代碼要少很多，使得其結構相應簡化，因此，在體積，造價，功耗，散熱，價格上都有優勢。

從以上的比較來看，RISC 與 CISC 各有優劣，而 RISC 的實用性則更強一些，應該是未來處理器架構的發展方向。但事實上，由於早期的很多軟體是根據 CISC 設計的，單純的 RISC 將無法兼容，此外，現代的 CISC 結構的 CPU 已經融合了很多 RISC 的成分，其性能差距已經越來越小，而複雜的指令可以提供更多的功能，這是程式設計所需要的，因此，CISC 與 RISC 的融合應該是未來的發展方向。

REFERENCES

1. Wikipedia RISC
2. Wikipedia CISC
3. Assembly language for Inter-based computers Kip R. Irvine