新世紀電腦架構：

支援物件導向技術的微處理器設計及製作

摘要

物件導向程式設計(OOP)是近幾年來程式設計的新觀念，是站在人性化的觀點思考程式的邏輯。現今使用的電腦大可分爲兩大類型:複雜指令集電腦(CISC)和精簡指令集電腦(RISC)，由於這些電腦架構對於物件導向技術並沒有作出支援，故此物件導向程式在此等電腦上運行的速度並不理想。另一方面，現今電腦架構對於物件保安的支援並不足夠，基於此等因素，新的電腦架構應運而生。高級指令集電腦(HISC)是一部綜合形電腦，它擴充了典型描述式電腦的設計，用以解決當前電腦架構的問題，它主要能提供 OOP 的支援，更佳的保安技術和多媒體應用程式的支援。從初步的模擬結果顯示，JAVA 程式在 HISC 上運行，比使用 JIT 編譯器在 SPARC 上運行要快上十倍以上。

在這篇文章中，我們主要介紹如何利用可編程式編輯邏輯閘陣列(FPGA)來製造一個使用高級指令集電腦設計的輔助處理器，並且使用這個輔助處理器系統去評估高級指令集電腦的實際性能和表現。在實驗中，我們會使用 Nallatech Ballynuey Virtex PCI Card 上的 FPGA 作爲實驗的基本。基於初步試驗關係，這次製作的 HISC 輔助處理器會集中於指令集和快速記憶體方面的製作及研究。

簡介

在以前，複雜指令集電腦(CISC)被廣泛使用。在複雜指令集電腦中，一個指令能作出數個低層次工序，好像記憶體存取，數學運算，或是地址計算。這些強大功能的指令，幫助了程式設計。程式設計員只需使用很少的指令，就能完成想要的動作。Intel x86、Motorla 68k、DEC VAX 和 IBM 360/370/390 用 CISC 設計的處理器。

正當電腦系統發展的同時，電腦指令所要求的動作變得更爲複雜。這不但增加了解譯指令所需使用的時間，更由於繁複的邏輯運算，大大增加了印模的面積。故此，另一種電腦架構，精簡指令集電腦(RISC)便被提倡使用。精簡指令集電腦的設計原則是使用簡單的指令去將繁複的指令取代，這樣的做法可加快電腦的運算速度和減少邏輯運算所使用的面積。精簡指令集電腦的例子有 SUN Microsystems

SPARC、MIPS R3000、Motorola 88100、Intel i860 和 IBM POWER。由於指令集簡單了，使用管線作業和超純量設計技術便更能提升精簡指令集電腦的速度。雖說精簡指令集電腦的表現不錯，但是，它仍是有不足之處的。由於精簡指令集電腦缺少了支援物件導向技術的架構，所以在精簡指令集電腦上運行物件導向程式的表現並不理想。

現今使用的兩大類型電腦架構，複雜指令集電腦和精簡指令集電腦，也同時表現出它們對物件導向技術支援的不足。爲此，我們更需要一個新的電腦架構，以補償現今電腦架構的不足。高級指令集電腦(HISC)就是基於這些問題，而設計出來的。高級指令集電腦源於典型描述式電腦的設計。它和現今電腦架構的最大分別，就是運算元描述組(Operand Descriptor, OD)的使用。透過運算元描述組，高級指令集電腦便可作出對物件導向技術支援，更佳的保安技術和多工處理的支援。運算元描述組記錄了每樣物件的屬性，例如物件型態、物件大小、向量值、存取權利、快速緩衝設定等等。電腦硬體透過這些物件的資料，便可以對物件導向程式作出直接支援。由於高級指令集電腦的指令大小是統一的，管線作業和超純量設計技術更可加於高級指令集電腦的設計中，以增強表現。從初步的模擬結果顯示，JAVA 程式在 HISC 上運行，比使用 JIT 編譯器在 SPARC 上運行要快上十陪以上。

在今次的計劃中，基於方士驤先生所提倡的高級指令集電腦，我們使用可編程式編輯邏輯閘陣列(FPGA)來製作一個 HISC 輔助處理器。爲了簡化實驗的過程，我們使用了在 Nallatech Ballynuey Virtex PCI Card 上的 Xilinx Virtex XCV-800-6 FPGA 去製作及設計一個 HISC 輔助處理器系統。然後，透過一些爲此系統設計的除錯程式，我們便可以對 HISC 系統進行測試和檢查。

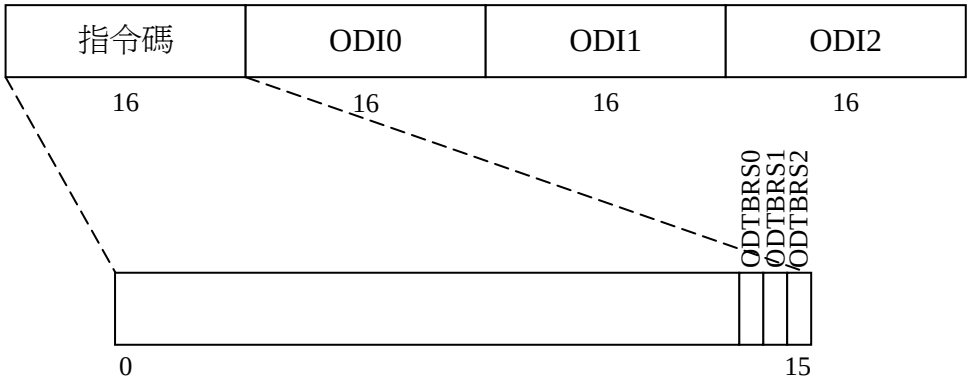
高級指令集電腦的架構

高級指令集電腦是一部綜合形電腦，它擴充了典型描述式電腦的設計以提供硬體對 OOP 的支援，更佳的保安技術和多媒體應用程式的支援。

指令格式

每一個 HISC 指令包含一個指令碼(Opcode)和三個運算元描述組指標(Operand Descriptor Indexes, ODIs)。運算元描述組指標會指向一個運算元描述組表格

(Operand Descriptor Table, ODT)。在 HISC 的設計中，儲存了兩個運算元描述組表格基暫存器(Operand Descriptor Table Base Registers, ODTBRs)以作運算元描述組表格的參考。設計中，使用一些特別的指令，就可以更新這兩個暫存器。另外，指令碼的第十三至第十五位元表示了三個運算元描述組指標將會使用那一個運算元描述組表格作爲參考基礎。



圖一・HISC 指令格式

運算元描述組格式

每一個運算元描述組記錄了以下的物件資料，物件地址、物件型態、物件大小、向量值、存取權利、快速緩衝設定及定址模式。在現在的設計中，我們定義了兩個定址模式，它們是直接定址模式和堆疊指標器(Stack Pointer)相對定址模式。在直接定址模式中，物件地址記錄了運算元在記憶體中的真實位址。而在堆疊指標器相對定址模式中，物件地址會被用作爲加上堆疊指標器的二補數(2's complement offset)彌補。

物件位址	物件型態	物件大小	向量值	存取權利	快速緩衝設定	定址模式	未定義
48	4	16	16	4	2	2	36

圖二・HISC 運算元描述組格式

定址模式	描述
00	直接定址模式
01	堆疊指標器相對定址模式
10	未定義
11	未定義

列表一・HISC 定址模式

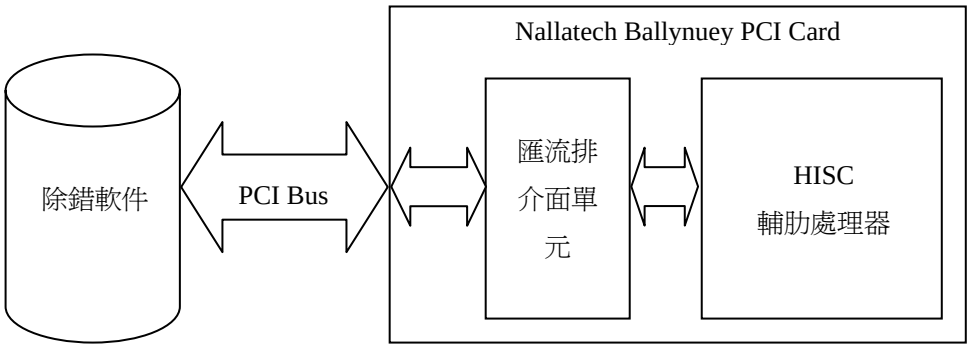
高級指令集電腦支援不同的資料型態(見列表二)。每一件物件的資料型態均記錄在運算元描述組的物件型態中。而物件大小則代表了該物件所佔用的位數。在現今這個階段，其它物件屬性並未完全被定義，故此，我們今次並不會談論到其它屬性的使用。

編碼	定義
0	資料型態未定義
1	邏輯及整數
2	IEEE 754 浮點數
3	二進碼十進數
4	字元字串
5	UCS-4 字元字串
6	位元字串
7	資料型態未定義
8	無號飽和定點數
9	有號飽和定點數
A	物件參考
B	功能及程序

列表二・HISC 物件型態

HISC 輔助處理器系統的設計

在這次的計劃中，我們會使用 Nallatech Ballynuey Virtex PCI Card 上的 Xilinx Virtex XCV 800-6 FPGA 去製作這個輔助處理器系統。這一張 Ballynuey Virtex PCI Card 提供了一隻 onboard FPGA 和四個 DIME 模組以作硬體設計。另外，它亦提供了一些軟件程序以作爲 PCI Card 的軟件介面。透過程序呼叫，我們可以經由 PCI Bus 寫入或讀取三十二位元的資料。圖三描繪了我們 HISC 輔助處理器系統的設計。



圖三。 HISC 輔肋處理器系統的設計

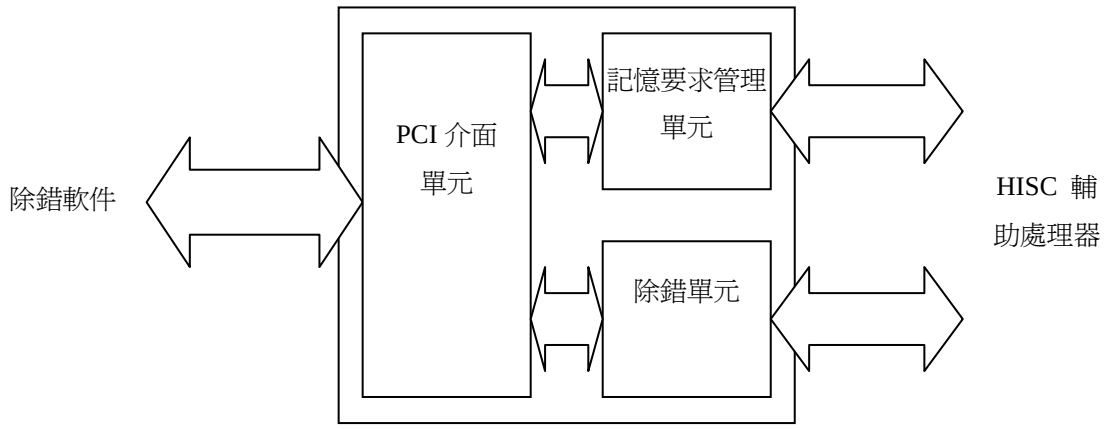
這個使用 PCI 介面的輔肋處理器系統包含了三個部份。它們是除錯軟件，匯流排介面單元，和 HISC 輔助處理器。在實際環境中，匯流排介面單元和 HISC 輔助處理器是位於 PCI Card 的 onboard FPGA 中的。

除錯軟件

這個除錯軟件的設計是用於監察 HISC 輔助處理器的功能和提供一個除錯環境。它同時亦扮演著系統上的主記憶體角色。透過這個軟件，不同的程式可以經由 PCI Bus 載入 Nallatech PCI Card 中。HISC 輔助處理器便會跟據程式的設計去運作。在程式執行中，一些除錯資訊會顯示在軟件的介面上以提供輔助處理器的最近狀況。

匯流排介面單元

匯流排介面單元扮演著中間人的角色，它提供了軟件和 HISC 輔助處理器之間的溝通渠道。在一方面，它會將 HISC 輔助處理器的記憶體要求通知除錯軟件。而另一方面，除錯軟件亦可以透過一些指令去要求匯流排介面單元給予它 HISC 輔助處理器的資料。匯流排介面單元可以分成三個模組，它們是 PCI 介面單元，記憶要求管理單元，及除錯單元。PCI 介面單元的作用是接收由 PCI Bus 傳來的指令，並對其進行解碼及作出適當的種作。而記憶要求管理單元則負責集合三個快取記憶體的要求，並通知 PCI 介面向軟件查詢。除錯單元的設計在於收集 HISC 輔助處理器的資料狀況。另外，我們可以進入逐步除錯模式以方便除錯。

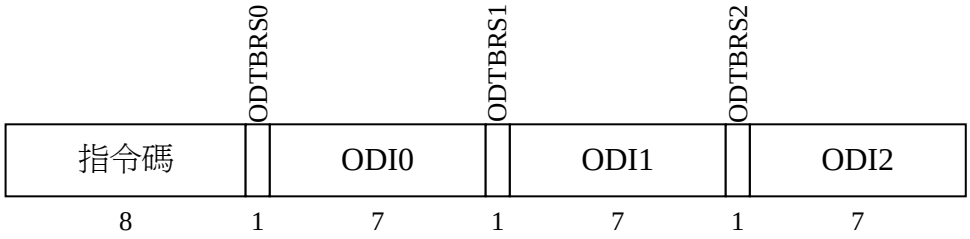


圖四・Bus 介面單元

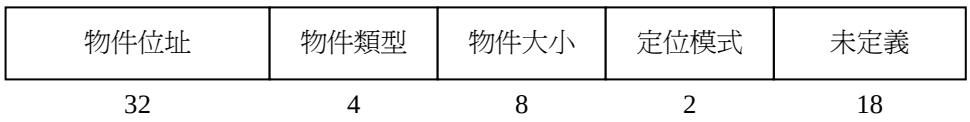
HISC 輔助處理器

HISC 原本是一個六十四位元的架構。但在 HISC 輔助處理器的設計中，我們將 HISC 架構簡化至三十二位元。原因是由於我們所使用的 Ballynuey PCI Card 只提供每次三十二位元的資料存取，而爲了簡化資料傳送的過程，我們有必要將 HISC 架構簡化成三十二位元。在這個 HISC 系統中，一個 word 代表了三十二個位元。

由於我們簡化了 HISC 輔助處理器的架構，所以我們要重組 HISC 的指令格式和運算元描述組格式。新的指令格式和運算元描述組格式已描繪在圖五及圖六。從圖中，我們可以看見有些物件屬性已在新的運算元描述組中移除。正如今次的計劃，只集中於在指令集及快取記憶體上的開發，所以其他 HISC 的特色，如物件存取控制，中斷，等等，並不會在今次的計劃中作研究及開發。

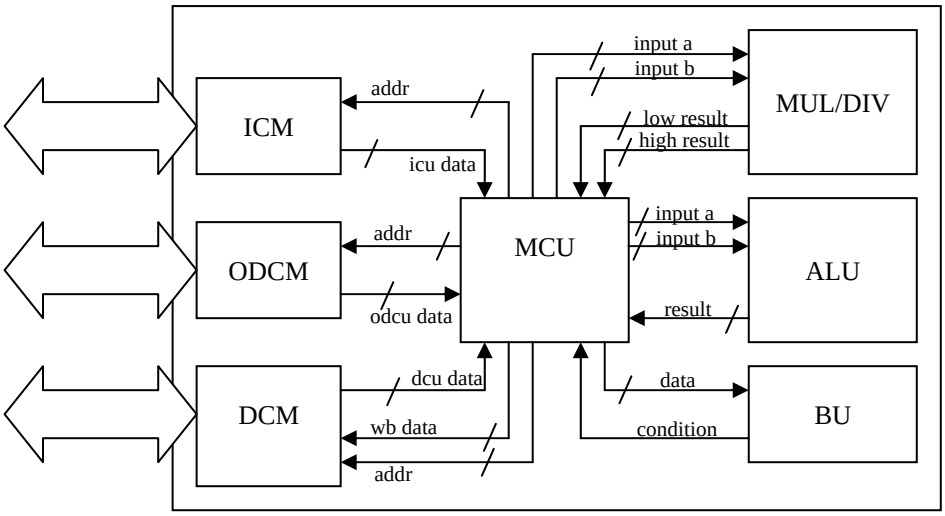


圖五・HISC 輔助處理器的指令格式



圖六・HISC 輔助處理器的運算元描述組格式

在 HISC 輔助處理器中包含了三個快取記憶體模組，主管單元(MCU)，乘/除單元(MUL/DIV)，算術及邏輯運算單元(ALU)及跳躍單元(BU)。



圖七・HISC 輔助處理器的架構

指令快取記憶體模組(Instruction Cache Module, ICM)

指令快取記憶體模組主要負責處理所有來自 MCU 的指令要求。在 HISC 輔助處理器中，指令快取記憶體會設計成 64KB 的直接映射快取記憶體。每一個 block 可以儲存四個 word，整個記憶體取多可以記錄 4k blocks。在 tag ram 中，存有一個叫 valid 的位元，它會指示出在快取記憶體中的資料是否正確。由於指令快取記憶體是用作記錄指令，所以資料取代的時候資料不會被寫回主記憶體中。

位元	16	128	1
Block	Tag	資料	valid
0			
1			
◦			
◦			
◦			
◦			
◦			
4095			

圖八・指令快取記憶體

運算元描述組快取記憶體模組(Operand Descriptor Cache Module, ODCM)

運算元描述組快取記憶體模組主要負責處理所有來自 MCU 的運算元描述組要求。它和指令快取記憶體模組的記計大同小異。運算元描述組快取記憶體可以儲存 256 blocks 的資料。而每一個 block 可以儲存二個運算元描述組(即四個 word)。

位元	20	128	1
block	Tag	資料	valid
0			
1			
。			
。			
。			
。			
。			
255			

圖九・運算元描述組快取記憶體

資料快取記憶體模組(Data Cache Module, DCM)

資料快取記憶體模組主要負責處理所有來自 MCU 的資料要求。在 HISC 輔助處理器的設計中，它會被製作成直接映射寫回快取記憶體。資料快取記憶體可以儲存 16K block 的資料，而每一個 block 可以儲存 4 個 word 的資料。另外，在 tag ram 中，除了 valid 位元外，另外加入了一個 dirty 位元。它的用途是記錄快取記憶體中的資料應否寫回主記憶體中。

位元	14	128	1	1
Block	tag	資料	dirty	valid
0				
1				
。				
。				
。				
。				
。				
16383				

圖十・資料快取記憶體

乘/除單元(Multiply/Divide Unit, MUL/DIV)

乘/除單元主要負責處理無號的乘數運算和除數運算。在乘數的運算中，我們使用了 2-bit Booth's Algorithm。而在除數的運算中，我們使用了 1-bit Non-Restoring Divide Algorithm。乘/除單元和 ALU 構成了 HISC 輔助處理器中的執行單元。

算術及邏輯運算單元(Arithmetic and Logic Unit, ALU)

算術及邏輯運算單元主要負責處理所有算術和邏輯上的運算。它是 HISC 輔助處理器執行單元的其中一部份。除了乘數和除數外，算術及邏輯運算單元負責其他的數學運算，例如 ADD、SUB、INC、DEC、PASS、AND、OR、XOR、等等。

跳躍單元(Branch Unit, BU)

跳躍單元主要負責處理檢查所有跳躍 condition。同時，它會通知 MCU 比較的結果。基本上，程式的流程是由 MCU 處理的，跳躍單元只負責比較結果。

主管單元(Main Control Unit, MCU)

主管單元是 HISC 輔助處理器的主要核心。它控制了整個機器的流程。在執行的時候，主管單元會經歷多個運作的階段。它們是指令提取，指令解碼，運算元描述組提取，運算元存取檢查，資料提取，執行及寫回。在每一個不同的階段，主管單元會通知其他模組去執行適當的動作。

HISC 輔助處理器系統的製作

在 HISC 輔助處理器系統中，它包含了軟體及硬體的製作。

軟體製作

系統中的軟體製作，主要是基於 Nallatech Ballynuey PCI Card 所提供的軟件程序而製作的。透過呼叫適當的程序，我們作出 PCI Bus 的資料傳輸。在我們的製作中，定義了一些指令。當我們傳送不同的指令去匯流排介面單元的時候，匯流排介面單元會作出相對的反應。所以，透過一些既定的指令，除錯軟件可以知道現在 HISC 輔助處理器的情況。系統中的除錯軟件，是用 Microsoft Visual C++ 6.0 來製作的。除了除錯外，軟件亦扮演著系統中的主記憶體角色。

硬體製作

在 HISC 輔助處理器系統中，硬體的製作包括了二個部份，匯流排介面單元和 HISC 輔助處理器。要在 Nallatech PCI Card 上作硬體設計，我們先要使用數位硬體描述語言(HDL)來描述要製作的硬體。而今次，我們使用了 VHDL 這種數位硬體描述語言來描述我們系統中的硬體。透過使用一些 EDA 工具，我們可以將程式轉化爲一個位元檔案。而使用 Nallatech 所提供的軟件，我們可以將這個位元檔載入 Ballynuey PCI Card 中。在製作中，我們的硬體會位於 PCI Card 上的 onboard Virtex FPGA。

我們使用了有限狀態機(Finite State Machine)的設計去製作系統上的硬體。要使用 VHDL 來描述有限狀態機，我們雖要使用三個程序(process)的描述方式。例如，

process(state) begin case (state) is when S0 => output <= '00'; when S1 => output <= '01'; when S2 => output <= '10'; when S3 => output <= '11'; when others => output <= '--'; end case; end process;	第一個程序定義了有限狀態機的輸出

process(state)
begin
case (state) is
when S0 => next_state <= S1;
when S1 => next_state <= S2;
when S2 => next_state <= S3;
when S3 => next_state <= S0;
when others => next_state <= S0;
end case;
end process;
	第二個程序則決定了下個狀態

process(clk， reset_l)
begin
if (reset_l = '0') then
state <= S0;
elsif (clk 'event and clk = '1') then
state <= next_state;
end if;
end process;
	而第三個程序描述了有限狀態機所需要的循序(sequential)邏輯

圖十一・三個程序的描述方式

我們基於以上的方法，製作了不同的硬體模組。

主管單元

我們使用了有限狀態機的設計去製作主管單元。在每一個不同的狀態下，它會控制不同的模組作出不同的反應。

乘/除單元

計算乘數，我們使用了 Modified Booth's Algorithm 去製作，而計算除數，我們則使用 Non-Restoring Algorithm 去製作。這個單元可以在十六個循環內作出三十二位元的乘數和三十二個循環內作出三十二位元的除數。我們使用了傳交握信號(Handshaking Signal)的設計去通知單元開始運作及完成計算後通知其他單元。

快取記憶體的製作

在 Virtex FPGA 裏，它儲有一些同步式的 onChip 記憶體我們使用了這些記憶體可以用作爲 HISC 輔助處理器的第一層快取記憶體。使用 Xilinx Core Generator 軟件，我們可以在製作中使用這些記憶體。

在現在這個階段，我們並未完全將整個指令集加入 HISC 輔助處理器中。其中一些指令，如物件管理的指令仍亦在研究階段。另外，浮點數字的運算亦未包括在今次的設計中。

系統測試

在 HISC 輔助處理器系統中，我們製作了一個除錯軟件，用以測試 HISC 輔助處理器的功能。我系統測試中，我們設計了一些測試向量(Test Vector)。透過使用除錯軟件，我們可以將測試向量放入 HISC 輔助處理器中並進行測試，所有測試結果顯示， HISC 輔助處理器的功態是正確的。

總結

正當物件導向技術漸??普遍，它將會是明天開發軟件的潮流。故此，我們非常需要一個能支援物件導向技術的硬體。在今次的計劃中，我們製作了一個 HISC 輔助處理器系統。透過這個系統，我們可以對高級指令集電腦架構進行測試及研究。我這個系統中，它包含了一個除錯軟件，匯流排介面單元，及 HISC 輔助處

理器。除錯軟件的開發，是爲了給予用家一個介面去對 HISC 輔助處理器作出監察。而匯流排介面單元，則提供了軟件和 HISC 輔助處理器之間的溝通橋樑。所以，透過這個系統，我們可以用最少的力量去對 HISC 輔助處理器作出研究。有了這個系統，我們將更能集中於對 HISC 架構的開發及研究。

在 HISC 架構中，仍有很多地方有待研究。在現在這個階段，我們所製作的只是一些對 OOP 作出支援的指令。其他 HISC 的特點，如物件保安，物件管理等等，都仍是在開發的階段。但基於我們所設計的 HISC 輔助處理器系統，我們將可以更容易地對其他方面，如 OOP 的全面支援，更佳的保安，及多媒體軟件的支援等等，作出更深入的研究。

未來發展方向

我們將會繼續改良 HISC 的指令。另外，物件處理，存取權利和快速緩衝設定的設計及製作將會增強對作業系統的支援。透過 HISC 輔助處理器系統的幫助，我們更可以比較一下現今的電腦系統和 HISC 系統的性能及表現。

參考

[1] Steve Heath. Microprocessor architectures and systems: RISC, CISC, and DSP. Jordan Hill, Oxford: Newnes, 1991.

[2] J.C. Heudin and C. Panetto. RISC Architectures. London : Chapman & Hall, 1992.

[3] Anthony S. Fong. “HISC: A High-level Instruction Set Computer”. In 7th European Simulation Symposium, pages 406-410. The Society for Computer Simulation, Oct 1995.

[4] Richard C. L. Li, Anthony S. Fong, Derek Pao. “Architecture Support of A Descriptor Computer on Object-Orientation”. In 13th ISCA International Conference on Computer and their Application. Honolulu, Hawaii USA, March 1998.

[5] V. Carl Hamacher, Zvonko G. Vranesic and Safwat G. Zaky. Computer Organization The Fourth Edition. McGraw-Hill Companies, Inc.