CHƢƠNG 1  
  
GIỚI THIỆU – CISC & RISC  
  
1.1. Giới thiệu chung  
Bài tập lớn này nhằm giúp chúng ta hiểu rõ tất cả những vấn đề liên quan đến thiết  
kế bộ vi xử lý theo hƣớng CISC và RISC.  
Báo cáo gồm 3 phần:  
Phần 1: Giới thiệu – CISC & RISC  
Phần 2: Kiến trúc – CISC & RISC

Phần 3: Ƣu điểm và nhƣợc điểm – CISC & RISC  
Phần 4: Tổng kết  
Và cuối cùng là các nguồn tài liệu tham khảo.  
  
1.2. Giới thiệu về CISC  
Những ngày đầu của ngành công nghiệp máy tính, trình biên dịch chƣa xuất hiện.  
Công việc lập trình đƣợc thực hiện hoặc bằng ngôn ngữ máy (mã nhị phân) hoặc bằng  
hợp ngữ. Để việc lập trình đơn giản, các VXL đƣợc thêm những lệnh có thể biểu diễn  
trực tiếp những cấu trúc của ngôn ngữ lập trình cấp cao. Lúc đó thiết kế phần cứng dễ  
hơn nhiều so với thiết kế trình dịch, vì thế mọi phức tạp đều dồn vào phần cứng.  
Một nguyên nhân khác thúc đẩy sự ra đời của những lệnh phức là sự thiếu thốn bộ  
nhớ. Do bộ nhớ quá nhỏ, do đó sẽ có lợi hơn nhiều nếu tăng mật độ tập trung thông tin  
trong mã lệnh. Khi mà mỗi byte bộ nhớ còn quá đắt, bộ nhớ chính của toàn bộ hệ  
thống ở thời kỳ này chỉ vài KB, ngành công nghiệp VXL bị thúc đẩy phải mã hóa thật  
cao mã lệnh, mã lệnh có thể có kích thƣớc thay đổi, một lệnh có thể thực hiện nhiều  
phép toán hoặc một lệnh có thể vừa chuyển dữ liệu vừa xử lý dữ liệu. Lúc đó việc đƣa  
ra một lệnh nén thật tốt đƣợc ƣu tiên hơn là đƣa ra một lệnh dễ giải mã.  
Lúc này bộ nhớ đƣợc sản xuất bằng công nghệ từ do đó nó không những đã ít mà  
còn chậm. Đây cũng là một lý do để tăng mật độ thông tin trong một mã lệnh. Một mã  
lệnh với nhiều thông tin sẽ giảm đƣợc rất nhiều lần phải truy xuất nguồn bộ nhớ chậm  
chạp này.  
2  
  
  
Những CPU thời kỳ này chứa ít thanh ghi vì những lý do sau:  
•  
  
•  
  
Một bit trong CPU bao giờ cũng mắc hơn rất nhiều so với một bit ở bộ nhớ ngoài.

Với công nghệ tích hợp ở thời kỳ này muốn có thêm thanh ghi bắt buộc phải có  
thêm vùng trống trên board hoặc trên chip.  
Một lƣợng lớn thanh ghi cũng sẽ cần một lƣợng lớn các bit trong mã lệnh để xác  
định các thanh ghi đó.  
  
Vì những lý do trên, những nhà thiết kế VXL cố gắng để mỗi lệnh có thể thực  
hiện càng nhiều chức năng càng tốt. Điều này dẫn đến một lệnh sẽ làm tất cả công việc  
nhƣ nạp 2 số cần cộng, cộng chúng lại, và cuối cùng lƣu trở lại vào bộ nhớ. Cũng lệnh  
đó lại có thể đọc một số từ thanh ghi và số còn lại từ bộ nhớ sau đó lƣu kết quả vào bộ  
nhớ. Khuynh hƣớng thiết kế VXL này đƣợc gọi là Complex Instruction Set Computer  
– CISC.  
CISC (Complex Instruction Set Computer) đƣợc nghĩ ra từ những năm 1960. Vào  
thời kỳ này, ngƣời ta nhận thấy các chƣơng trình dịch khó dùng các thanh ghi, rằng  
các vi lệnh đƣợc thực hiện nhanh hơn các lệnh và cần thiết phải làm giảm độ dài các  
chƣơng trình. Các đặc tính nầy khiến ngƣời ta ƣu tiên chọn các kiểu ô nhớ - ô nhớ và ô  
nhớ - thanh ghi, với những lệnh phức tạp và dùng nhiều kiểu định vị. Điều này dẫn tới  
việc các lệnh có chiều dài thay đổi và nhƣ thế thì dùng bộ điều khiển vi chƣơng trình  
là hiệu quả nhất.  
Bảng 1 cho các đặc tính của vài máy CISC tiêu biểu. Ta nhận thấy cả ba máy đều  
có điểm chung là có nhiều lệnh, các lệnh có chiều dài thay đổi. Nhiều cách thực hiện  
lệnh và nhiều vi chƣơng trình đƣợc dùng.  
Tiến bộ trong lãnh vực mạch kết (IC) và kỹ thuật dịch chƣơng trình làm cho các  
nhận định trƣớc đây phải đƣợc xem xét lại, nhất là khi đã có một khảo sát định lƣợng  
về việc dùng tập lệnh các máy CISC.  
  
Bảng 1. Đặc tính vài máy CISC  
3  
  
  
Sau một thời gian ngƣời ta nhận thấy rất nhiều tính năng trong các bộ VXL vốn

đƣợc thiết kế nhằm giúp công việc lập trình trở nên dễ dàng hơn lại thƣờng bị các phần  
mềm bỏ sót. Những tính năng này thông thƣờng cần vài chu kỳ máy để thực thi. Cộng  
thêm sự cách biệt về hiệu suất giữa các CPU và bộ nhớ chính đã dẫn đến nhiều kỹ  
thuật hoặc nhằm tổ chức lại quá trình thực thi trong bộ xử lý hoặc nhằm giảm bớt số  
lần truy xuất bộ nhớ.  
  
1.3. Giới thiệu về RISC  
Đến những năm cuối của thập niên 1970, các nhà nghiên cứu của IBM (và cả một  
số dự án khác) đã chứng minh rằng phần lớn các phƣơng pháp đánh địa chỉ trực giao  
thƣờng bị các chƣơng trình bỏ qua. Đây chính là kết quả không mong đợi do sử dụng  
các trình biên dịch cấp cao thay vì sử dụng hợp ngữ. Các trình dịch tại thời điểm đó  
không đủ khả năng để tận dụng hết tính năng của các bộ VXL CISC; chủ yếu là do sự  
khó khăn trong thiết kế trình dịch. Trình biên dịch càng trở nên phổ biến thì các tính  
năng này lại càng bị bỏ quên.  
Một nghiên cứu khác cũng chỉ ra rằng những tính năng này ít đƣợc dùng vì thực ra  
chúng đƣợc thực thi chậm hơn một nhóm lệnh cùng thực hiện tác vụ đó. Đây giống  
nhƣ một nghịch lý của quá trình thiết kế VXL, ngƣời thiết kế không có đủ thời gian để  
tối ƣu cho tất cả các lệnh, do đó họ chỉ chú trọng đến những lệnh thƣờng đƣợc sử dụng  
nhiều nhất.  
Cũng trong thời gian này, CPU bắt đầu hoạt động nhanh hơn bộ nhớ. Thậm chí  
trong thập niên 1970, ngƣời ta cho rằng điều này sẽ còn tiếp tục không dƣới 1 thập  
niên nữa, và tới lúc đó CPU sẽ nhanh hơn bộ nhớ hàng chục tới hàng trăm lần. Có lẽ  
đã đến lúc CPU cần thêm nhiều thanh ghi (sau này gọi là cache) để có thể hoạt động ở  
tốc độ cao hơn. Những thanh ghi và bộ nhớ đệm mới sẽ cần khoảng trống trên bo  
mạch hoặc trên chip đƣợc tạo ra nếu giảm sự phức tạp của CPU.  
Tới lúc này, một phần đóng góp cho kiến trúc RISC đến từ thực tế đo đạc những  
chƣơng trình trong thế giới thực. Andrew Tanenbaum từ tổng kết rất nhiều đo đạc này  
đã chỉ ra rằng hầu hết những CPU lúc bấy giờ đều đƣợc thiết kế thừa quá mức. Ví dụ,  
ông cho rằng 98% các hằng hoàn toàn có thể biểu diễn bằng 13 bit, trong khi đó các  
CPU đƣợc thiết kế theo bội số của 8 thƣờng là 8, 16 hoặc 32. Do đó nếu CPU cho

phép các hằng đƣợc lƣu trong những bit dƣ của mã lệnh sẽ làm giảm đi rất nhiều lần  
truy xuất bộ nhớ. Thay vì phải đọc từ bộ nhớ hay từ thanh ghi, các hằng đã ở ngay đó  
khi CPU cần, vì thế quá trình thực thi sẽ nhanh hơn. Tất nhiên điều lại này yêu cầu mã  
lệnh phải thật nhỏ để những lệnh 32 bit có thể chứa đƣợc những hằng tƣơng đối lớn.  
  
4  
  
  
Những chƣơng trình trong thực tế thƣờng tốn phần lớn thời gian để thực hiện một  
số tác vụ đơn giản, do đó một số nhà nghiên cứu hƣớng tới việc tối ƣu hóa những tác  
  
vụ này. Do xung nhịp (clock rate) của CPU bị giới hạn bởi thời gian thực hiện lệnh  
chậm nhất, nên nếu tối ƣu lệnh này (có thể bằng cách giảm số phƣơng pháp đánh địa  
chỉ mà nó hỗ trợ) sẽ khiến cho toàn bộ tập lệnh đƣợc thực thi nhanh hơn nhiều. Mục  
tiêu của RISC chính là đơn giản hóa các lệnh, để mỗi lệnh có thể đƣợc thực thi chỉ  
trong 1 chu kỳ máy. Việc tập trung đơn giản hóa các lệnh đã cho ra đời các loại "Máy  
tính với tập lệnh đƣợc đơn giản hóa" - RISC.  
Bảng 2 Đặc tính vài máy RISC  
Điểm khác biệt thực sự giữa RISC so với CISC là nguyên tắc thực hiện mọi thứ  
trong các thanh ghi, đọc và lƣu dữ liệu vào các thanh ghi. Do đó để tránh hiểu lầm  
nhiều nhà nghiên cứu thích dùng thuật ngữ load-store.  
Qua thời gian, các kỹ thuật thiết kế cũ đƣợc gọi là CISC (Complex Instruction Set  
Computer – Máy tính với tập lệnh phức tạp), dù rằng thuật ngữ này thƣờng chỉ đƣợc  
dùng với mục đích so sánh.  
Giờ đây để thực hiện cùng một công việc, chƣơng trình đƣợc viết với những lệnh  
đơn giản thay vì với một lệnh phức tạp, tổng số các lệnh phải đọc từ bộ nhớ nhiều hơn  
do đó cũng mất nhiều thời gian hơn. Tại thời điểm đó ngƣời ta chƣa biết khuyết điểm  
này có còn đảm bảo sự ƣu việt hơn về hiệu suất của RISC hay không, và hầu nhƣ đó  
cũng đã là một cuộc chiến dai dẳng về khái niệm RISC.

5  
  
  
CHƢƠNG 2  
  
KIẾN TRÚC – CISC & RISC  
2.1. Tổng quan  
  
Hiệu suất máy tính luôn là vấn đề đƣợc quan tâm hàng đầu đối với việc phát triển  
ngành công nghiệp máy tính. Để có một hiệu suất tốt, bộ xử lý trung tâm (CPU Central Processing Unit) cần đƣợc thiết kế để tính toán, làm việc từ kiến trúc tập lệnh  
(ISA- Instruction Set Architecture) một cách hiệu quả. Hiệu suất máy tính có thể đƣợc  
đo thông qua thời gian chạy của chƣơng trình.  
  
Dựa vào biểu thức trên, có thể thấy rằng để tăng hiệu suất của máy tính (giảm  
thời gian thực thi chƣơng trình) có 2 cách thực hiện:  
6  
  
  
• Giảm số câu lệnh trong một chƣơng trình. Điều này đƣợc thực hiện bằng cách  
thiết kế tập lệnh sao cho mỗi lệnh có thể thực thi đƣợc những tính toán phức tạp, đây  
là hƣớng tiếp cận của kiến trúc CISC. CISC có khả năng thực hiện việc thực thi các  
phép toán về chế độ địa chỉ (addressing mode) hoặc các tính toán nhiều bƣớc trong  
một tập lệnh. Đây là thiết kế của CPU mà một câu lệnh thực hiện nhiều phép toán ở  
mức độ thấp nhƣ lƣu trữ vào bộ nhớ, các phép toán số học và lấy dữ liệu từ bộ nhớ.  
Intel 80486 đƣợc thiết kế theo cách này.  
• Giảm chu kì (cycle) thực thi lệnh: các lệnh máy sẽ đƣợc tối giản, chỉ thực hiện  
những công việc đơn giản để không tiêu tốn nhiều chu kì, đây là kiến trúc RISC. RISC  
là một chiến lƣợc thiết kế CPU dựa trên nhận thức rằng đơn giản hóa tập lệnh cho hiệu  
suất cao hơn khi kết hợp với một kiến trúc vi xử có khả năng thực hiện các câu lệnh  
bằng một chu kì. Đa số các bộ xử lý hiện nay đƣợc thiết kế theo mô hình này.

Phần tiếp theo sẽ giới thiệu chi tiết về kiến trúc CISC và RISC  
  
2.2. Kiến trúc CISC  
Hƣớng tiếp cận của mô hình CISC là cố gắng giảm thiểu số lƣợng câu lệnh của  
chƣơng trình, hy sinh số chu kỳ của mỗi câu lệnh, điều này là dễ hiểu vì câu lệnh càng  
xử lý phức tạp thì tốc độ thực thi câu lệnh đó càng lâu. Những máy tính dựa trên kiến  
trúc CISC đƣợc thiết kế để giảm chi phí bộ nhớ. Trong nhiều thập kỉ trƣớc đây, khi mà  
chi phí bộ nhớ cực kì đắt đỏ, nếu chỉ sử dụng những câu lệnh đơn giản thì không gian  
lƣu trữ của chƣơng trình trên bộ nhớ sẽ nhiều hơn, tăng chi phí bộ nhớ. Để giải quyết  
vấn đề này, số lƣợng câu lệnh trên chƣơng trình có thể đƣợc giảm bằng cách gom một  
số lệnh đơn giản vào một câu lệnh duy nhất, do đó làm cho các câu lệnh phức tạp hơn,  
bù lại giảm thiểu đƣợc chi phi bộ nhớ. Một lợi ích nữa của kiến trúc CISC là giúp cho  
các trình biên dịch không cần phải thiết kế các lệnh quá cụ thể, bởi đã có những câu  
lệnh phức tạp.  
  
7  
  
  
ĐẶC ĐIỂM CỦA KIẾN TRÚC CISC  
- Việc giải mã câu lệnh là phức tạp.  
- Một lệnh đƣợc yêu cầu hỗ trợ nhiều chế độ địa chỉ.  
- Không gian chip nhỏ là đủ cho các thanh ghi mục đích chung của các câu lệnh  
đƣợc tính toán trực tiếp vào bộ nhớ.  
- Các thiết kế CISC khác nhau thiết lập hai thanh ghi đặc biệt cho con trỏ ngăn  
xếp (stack pointer), xử lý ngắt (interupt), ...  
Các phần cứng sử dụng kiến trúc CISC:  
IBM System/360 (S / 360) - là một mainframe trong đại gia đình máy tính của  
IBM, đƣợc công bố vào ngày 07 tháng 4 năm 1964.  
  
VAX 11/780 - thiết kế CISC là một bộ xử lý 32-bit và hỗ trợ rất nhiều trong chế

độ địa chỉ (addressing mode) và lệnh máy, là sản phẩm của Digital Equipment  
Corporation.  
  
Intel 80486 - đƣợc đƣa ra trong năm 1989 và nó là một bộ xử lý CISC, trong đó  
các câu lệnh có độ dài biến đổi và có tổng cộng 235 câu lệnh.  
8  
  
  
2.3. Kiến trúc RISC  
  
RISC (Reduced Instruction Set Computer) đƣợc sử dụng trong các thiết bị cầm tay  
do hiệu quả năng lƣợng của nó. Ví dụ, Apple iPod và Nintendo DS. RISC là một loại  
kiến trúc bộ vi xử lý sử dụng bộ tối ƣu hóa cao của tập lệnh. RISC làm điều ngƣợc lại,  
giảm chu kỳ mỗi tập lệnh tại các chi phí của số tập lệnh trên mỗi chƣơng trình  
Pipelining là một trong những tính năng độc đáo của RISC. Nó đƣợc thực hiện bằng  
cách lồng ghép việc thực hiện một số tập lệnh trong một mẫu pipeline. Nó có lợi thế là  
hiệu suất cao hơn so với CISC.  
  
Bộ vi xử lý RISC gồm các tập lệnh đơn giản và đƣợc thực hiện trong một chu kỳ  
đồng hồ (clock cycel). RISC có ƣu điểm là tốn ít câu lệnh, do đó giảm dung lƣợng lƣu  
trữ trên RAM, ngoài ra còn giúp trình biên dịch không phải xử lý những câu lệnh phức  
tạp.  
  
9  
  
  
ĐẶC ĐIỂM KIẾN TRÚC RISC  
-  
  
Tập lệnh đơn giản đƣợc sử dụng trong kiến trúc RISC.

-  
  
RISC giúp đỡ và hỗ trợ một số kiểu dữ liệu đơn giản và tổng hợp các loại dữ  
  
-  
  
liệu phức tạp.  
RISC sử dụng các phƣơng thức giải quyết đơn giản và cố định chiều dài tập  
  
-  
  
lệnh để pipelining.  
RISC cho phép bất kỳ thanh ghi nào cũng có thể sử dụng trong bất kỳ hoàn  
  
-  
  
cảnh nào.  
Tốn một chu kỳ thời gian thực thi.  
Số lƣợng công việc mà máy tính có thể thực hiện đƣợc giảm bằng cách chia  
cho tập lệnh "LOAD" và “STORE".  
RISC chứa với số lƣợng lớn thanh ghi để ngăn chặn số lƣợng khác nhau của  
các tƣơng tác với bộ nhớ.  
Trong RISC, Pipelining là dễ dàng nhƣ việc thực hiện tất cả các lệnh sẽ đƣợc  
thực hiện trong một khoảng thời gian thống nhất. Ví dụ: 1 lần click chuột  
Trong RISC, RAM đƣợc yêu cần nhiều hơn là cần thiết để lƣu trữ hƣớng dẫn  
cấp lắp ráp.  
Giảm tập lệnh cần ít các transitor bán dẫn trong RISC.  
RISC sử dụng mô hình bộ nhớ Harvard (Harvard Kiến trúc).  
Một trình biên dịch đƣợc sử dụng để thực hiện các hoạt động chuyển đổi tức là

để chuyển đổi một tập lệnh ngôn ngữ cấp cao vào mã hình thức của nó.  
  
2.4. So sánh kiến trúc RISC vs CISC  
Điểm khác biệt chính của hai kiến trúc này là số lƣợng lệnh (instructions) và độ  
phƣc tạp của nó.  
  
10  
  
  
Thiết kế CISC liên quan đến những kiến trúc rất phức tạp, bao gồm số lƣợng lớn  
tập lệnh và kiểu địa chỉ, trong khi thiết kế RISC liên quan đến những tập lệnh đƣợc  
đơn giản hóa và biến đổi theo yêu cầu thực tế của chƣơng trình ngƣời dùng.  
  
Phép nhân hai số trong bộ nhớ: phần này trình bày sự khác biệt cơ bản giữa  
RISC và CISC qua ví dụ cụ thể:  
  
Giản đồ trên giới thiệu mô hình bộ nhớ của một máy tính phổ thông. Bộ nhớ chính  
đƣợc chia thành các vùng, đƣợc đánh số từ row\_1 : column\_1 đến row\_5 : column\_4.  
Dữ liệu đƣợc nạp (load) vào các thanh ghi.  
Đơn vị thực thi có trách nhiệm thực hiện mọi tính toán. Tuy nhiên, đơn vị thực thi  
chỉ có thể thực hiện với các dữ liệu đƣợc nạp vào các thanh ghi.  
Bài toán đạt ra tình kết quả phép nhân hai số, một số đƣợc lƣu ở vị trí 2:3 và một  
số đƣợc lƣu ở 5:2, sau đó lƣu kết quả lại vào 2:3  
11  
  
  
a) Phƣơng pháp tiếp cận CISC  
Mục đích chính của kĩ thuật CISC là hoàn thành công việc với số lƣợng  
dòng assembly ít nhất có thể. Điều này thành công bằng việc xây dựng bộ xử lý  
phần cứng có khả năng hiểu và thực hiện một chuỗi các phép toán. Bài toán trên

có thể hoàn thành chỉ bằng 1 lệnh:  
MULT 2:3, 5:2  
Khi thực thi, lệnh này sẽ nạp hai số vào hai thanh ghi riêng biệt, nhân hai  
toán hạng trong đơn vị thực thi, sau đó lƣu kết quả vào thanh ghi thích hợp.  
Lệnh MULT đƣợc gọi là tập lệnh phức tạp (complex instruction). Nó chạy  
trực tiếp trên bộ nhớ của máy tính và không yêu cầu ngƣời lập trình phải gọi lệnh  
nạp hay lƣu trữ nào hết, và gần giống nhƣ câu lệnh trong ngôn ngữ lập trình cấp  
cao. Cụ thể, nếu biến a biểu diễn giá trị 2:3 và biến b biểu diễn cho 5:2, câu lệnh  
trong C tƣơng đƣơng là a = a \* b.  
Lợi ích chính của hệ thống này là trình biên dịch chỉ cần làm rất ít để dịch  
từ ngôn ngữ cấp cao sang assembly, do code tƣơng đối ngắn, rất ít RAM cần cùng  
để lƣu trữ tập lệnh. Do đó nổi bất dùng để xậy dựng tập lệnh phức tạp trực tiếp  
trên phần cứng.  
b) Phƣơng pháp tiếp cận RISC  
Bộ xử lý RISC chỉ dùng các lệnh đơn giản đƣợc thực thi trong trong một  
clock cycle. Nhƣ vậy, lệnh “MULT” mô tả ở trên có thể đƣợc chi thành ba lệnh:  
LOAD – chuyển dữ liệu từ bộ nhớ vào thanh ghi  
PROD – tìm tích của các toán hạng đƣợc lƣu trong thanh ghi  
STORE – chuyển dữ liệu từ thanh ghi vào bộ nhớ  
Để thực hiện bài toán đặt ra, cần thực hiện 4 dòng hợp ngữ:  
LOAD A, 2:3  
LOAD B, 5:2  
PROD A, B  
STORE 2:3, A  
Đầu tiên, phƣơng pháp này có vẻ kém hiệu quả trong việc thực hiện phép tính, do  
cần nhiều dòng code hơn, nhiều RAM cần để lƣu những lệnh assembly. Trình biên  
dịch cũng cần thực hiên nhiều hơn để dịch từ ngôn ngữ cấp cao sang assembly.  
12

Tuy nhiên, do mỗi lệnh chỉ yêu cầu 1 clock cycle để thực thi, toàn bộ chƣơng trình  
sẽ đƣợc thực thi với lƣợng thời gian xấp xỉ với multi-cycle MULT trong CISC. RISC  
“reduced instructions” yêu cầu ít transistors trên phần cứng hơn, và dành nhiều chỗ  
cho nhiều thanh ghi dùng chung hơn.  
Việc tách riêng hai lệnh LOAD và STORE thực ra giảm công việc mà máy tính  
cần làm. Sau khi lệnh MULT trong CISC đƣợc thực thi, bộ xử lý tự động xóa thanh  
ghi. Nếu một toán hạng cần dùng cho một tính toán khác, bộ xử lý cần nạp lại toán  
hạng đó xuống thanh ghi. Nhƣng trong RISC, toán hạng vẫn còn trên thanh ghi cho tới  
ghi một giá trị khác đƣợc nạp vào thanh ghi đó  
  
13  
  
  
CHƢƠNG 3  
  
ƢU ĐIỂM VÀ NHƢỢC ĐIỂM  
3.1. Ƣu điểm  
a) CISC:  
- Microprogramming là ngôn ngữ assembly đơn giản để thực thi, và ít tốn kém  
hơn bao bọc đơn vị điều khiển.  
- Microprogramming dễ dàng các lệnh mới cho phép ngƣời thiết kế phát triển  
các máy CISC nhanh chóng phù hợp.  
- Mỗi lệnh đƣợc hoàn thành tốt hơn, ít lệnh cần dùng để thực hiện một công  
việc đƣợc đƣa ra.  
b) RISC:  
- Có một tập lệnh, nên trình biên dịch các ngôn ngữ cấp cao có tạo ra code hiệu  
quả hơn.  
- Cho phép sử dụng tự do các khoảng trống trên bộ vi xử lý (microprocessor) vì  
tính đơn giản của nó.  
- Nhiều bộ điều khiển dùng thanh ghi để truyền đối số và giữ các biến local.

- Các hàm trong RISC chỉ dùng ít tham số, bộ xử lý RISC không thể dùng các  
lệnh gọi, và hơn nữa, dùng lệnh có chiều dài cố định dễ dàng dùng pipeline.  
- Cần rất ít số lƣợng định dạng, lệnh và kiểu địa chỉ.  
  
3.2. Nhƣợc điểm  
a) CISC:  
- Hiệu suất của máy tính giảm do số lƣợng clock time của các lệnh không giống  
nhau.  
- Chỉ khoảng 20% các lệnh hiện tại thƣờng đƣợc dùng.  
- Conditional code đƣợc cài đặt bởi lệnh CISC ảnh hƣởng đến các lệnh cần thời  
gian co việc cài đặt này, và lệnh tiếp theo thay đổi các bit điều kiện (conditional  
code bits), do đó trình biên dịch cần kiểm tra bit điều kiện trƣớc.  
  
14  
  
  
b) RISC:  
- Gần nhƣ hiệu suất của bộ xử lý RISC phụ thuộc vào ngƣời lập trình hoặc trình  
biên dịch, kiến thức về compiler đóng vai trò quan trọng trong việc đổi từ CISC  
code sang RISC code.  
- Trong khi sắp xếp lại CISC code sang RISC code, kích thƣớc code sẽ tăng lên  
đáng kể. Và chất lƣợng của code mở rộng sẽ lại phụ thuộc vào compiler, và tập  
lệnh của máy.  
- 1st level cache của bộ xử lý RISC là một nhƣợc điểm. Bộ xử lý có đệm trên  
chính chip của nó. Để cung cấp cho các lệnh, đòi hỏi cần có hệ thống bộ nhớ  
thật nhanh.  
  
15

CHƢƠNG 4  
  
TỔNG KẾT  
4.1. Tổng kết  
Bảng tóm tắt một số điểm khác biệt chính giữa RISC và CISC  
CISC  
Nhấn mạnh trên phần cứng  
  
RISC  
Nhấn mạnh trên phần mềm  
  
Cycles per second (clock rate) cao, kích Cycles per second (clock rate) thấp, kích  
thƣớc code nhỏ  
thƣớc code lớn  
Transistors đƣợc dùng để lƣu trữ những Nhiều transistors hơn đƣợc dùng cho các  
tập lệnh phức tạp  
thanh ghi bộ nhớ  
Memory-to-memory: LOAD và STORE Register-to-register: LOAD và STORE là  
đƣợc thực hiện từ các lệnh  
các lệnh độc lập  
Multi-clock, các lệnh phức tạp  
  
Single-clock, đơn giản hóa các lệnh  
  
Kiến trúc RISC và CISC đƣợc phát triển nhƣ một thử nghiệm để bao phủ khoảng  
cách ngữ nghĩa (semantic gap):  
  
Figure 2. Semantic Gap  
Mới mục đích tăng hiệu suất phát triển phần mềm, rất nhiều những ngôn ngữ trừu  
tƣợng, súc tích bậc cao nhƣ C, C++, Java… đã đƣợc xây dựng. Với sự phát triển đó

khoảng cách về ngữ nghĩa cũng tăng lên. Để biên dịch hiệu quả các ngôn từ cấp cao,  
CISC và RISC là hai lựa chọn.