國立臺灣科技大學答案卷

National Taiwan University of Science and Technology Answer Sheet

評分 Score	教師簽章 Signature of Lecturer
90	

姓名/Name 張恒豪 學號/Student ID B11002110 班級/Class 四電子二乙

科目/Course title 計算機組織 日期/Date 111. 11. 4

記分欄 從此處開始寫起。試卷用紙務須節用，非經主試認可不得續用其他紙張作答。/Please write from here.

1. ISA 是連接軟體和硬體的介面，ISA 定義了各種指令的類型和結構。

+10 下圖所隱含的意思是如果將 ISA 設計得很複雜，那會造成雖然程式好寫，但要實作出每種指令相對的電路卻會比較困難。

2. 因為多核心的優勢在於能將指令平行化執行，但是程式卻無法做到完全的指令平行化，例如：

+10

1. ld x23, x24
2. add x23, x23, x24
3. lw x25, 0(23)
4. add x23, x25, x23 (B上方兩組箭頭同時執行)

如果同時執行 1. 2 和 3. 4 行就會發生衝突，因為 2 和 4 行都要寫入值到 x23，而在執行第三行時 A 的值其實取決於前面兩行的運算結果，所以如果 1. 3 行同時執行而沒有先執行 1. 2 行再執行第三行，程式就不會從正確的記憶體位址取出資料複製給 x25。

另外，指令平行化還需要顧慮到每個核心的工作負載是否平衡，如果每個核心所分配到的負載不均，平行化所帶來的執行效益也會降低。另外數個核心間也要透過即時的交換資料來確保變數的值是否有被更新過，這也需要耗費額外的時間來執行。由以上原因可知核心數和效能的提升並不成正比。

3. 因為這種指令會減少程式到記憶體抓取值的次數，例如將 x22 加 3 的動作，可以用下列程式表示：

+10

1. ld x21, 0(x5)
2. add x22, x22, x21

此記憶體位址存放數值 3

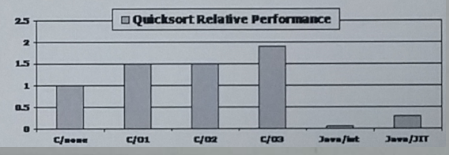
v.s.

1. add x22, x22, 3

如果用左側兩行將 x22 加 3，會需要將 3 這個數值從記憶體取出再做運算，但右側的程式不用，所以節省了從記憶體載入要運算的數值的時間。而且像這種將一個變數加上一個小數值是很常見的，像是 for 迴圈每次執行完的 i++ 靠右側程式的方法達成，就會比用左側程式來完成省下很多時間。因此許多 ISA 才會內建 immediate 格式指令。

因為透過將常數的欄位固定，可以讓電路在執行不同種指令時不需要再去依據指令來用多工器選擇相應的常數欄位，從而簡化電路，減短執行時間。

4. (10%) Please comment on the following figure.



記分欄

4 由此表可看出不同的編譯器對於同一種演算法的優化效果不同。換句話說，編譯器對於演算法是有敏感度的。也可看出直譯器和編譯器也會對演算法執行效能產生影響。 +8
(Java/Int) (Java JIT) Java vs. C ?

也可以看出不同的程式語言在執行同一個演算法時的執行效率會有所不同。這是因為不同的程式語言在編譯時所用的 Compiler 不同，編譯方法也不同，而造成編譯出的指令類型有所差異，因而造成執行效率有所不同。

5. 因為 Intel 所採用的 x86 指令集架構很複雜，造成指令的執行程序繁瑣且難以優化。所以 Intel 透過內部硬體電路將 x86 指令集轉換成較精簡的指令來讓執行(指行)的相對應硬體電路得以縮小規模進而減少延遲和更易優化來達成與 RISC 指令可比拼的效能。 +8
每種指令

6. Basic Block 是程式碼(連續)的某些區塊，在一個 Basic Block 中沒有任何跳出該區塊或返回該區塊的指令。 +8
因為 compiler 會特別去優化 Basic Block 中的程式碼。 +10

7. 因為在比大小的時候會比較快。正數不會比負數小，所以比大小先比正負號，如果正負不同就能直接得出結果，再來如果正負號相同，能再依指數的次方大小來判斷誰大誰小，如果正負和指數都相同才會從 fraction 來比較兩數的大小。這樣可以用最少的步驟得出結果，增加比大小的速度。 +8
bias ?

給 exponent 加 bias 是因為這樣在比次方大小的時候就不用判斷次方的正負號，可以直接當無號數比次方大小

8. 因為存放至記憶體內部的資料需要有固定的格式，才能確保下次取出的時候能得到和以前存入時相符的值。 +3
(數值)

9. 因為如果用 IC 來判斷效能，那麼如果一支程式編譯出來是 IC 小 CPI 大，那比較出的效能就會比 IC 大 CPI 小來得好然而效能是執行時間的反比，而執行時間 = IC × CPI × T，所以 IC 小並不表示 IC × CPI × T 的乘積會小，而其他參數本身也無法直接對應到和其他參數相乘得出的結果(執行時間)，因此不可以用單一或部分的 CPU execution time 的參數來判定效能。 +10
個別 或部分 個別

10. PC-relative addressing 是藉由程式跳躍前的 PC 值加上一個 offset 值來得出要跳躍到的目標位址。例如： +15

```
0x20    add   x21, x22, x23
0x24    sub   x24, x25, x22
PC→0x28 beq   x21, x24, Label1      Label1 = 0x34 - PC
0x32    ld    x21, 0(x3)            如果要跳, PC = PC + Label1
0x34 Label1:
```

國立臺灣科技大學答案卷

National Taiwan University of Science and Technology Answer Sheet

評分 Score	教師簽章 Signature of Lecturer

姓名/Name 張恆豪 學號/Student ID B11002110 班級/Class 四電子二乙

科目/Course title 計算機組織 日期/Date 111.11.4

記分欄 從此處開始寫起。試卷用紙務須節用，非經主試認可不得續用其他紙張作答。/Please write from here.

11. 因為在 2002 年以後，頻率已經無法有跳躍性的增長，像是 1996~2002 間兩代是頻率從 300 MHz 提升到 500 MHz，這兩代的頻率比值是 $\frac{500}{300}=\frac{5}{3}$，然而到 2002 年時上一代到下一代平均有 1GHz 的增長，像 Athlon 到 Athlon 64 的 2.9GHz 到 3.0GHz，它們的頻率比值是 $\frac{30}{29}$，明顯不如 1996~2002，而效能和執行時間成反比，執行時間和時脈週期成正比，因此效能會和工作頻率成正比，所以藉由兩代間的頻率比值可以看出 2002 以後兩代頻率比值無法像 1996~2002 那麼大，所以效能進步趨緩。

從單核轉到多核是因為電腦的 power (牽涉到散熱) 存在限制，而 power 與工作頻率成正比 ($P=CV^2f$)，所以想透過持續增加頻率來達成效能增長是不可能的，所以只好透過增加核心數來提升 throughput 來提升效能，但是堆疊核心數增加效能的效果沒比增加頻率 (減少 response 同時增加 throughput) 來得好，所以進步幅度減緩。

(增加 throughput)

why

增加 throughput 所帶來的效能增長會因為程式的指令間存在相依性，沒辦法完全平行化所受到限制。所以其帶來的效能增長不會比減少 response time 來的多。所以利用堆疊核心數所帶來的效能提升不會比增加頻率所帶來的效能提升多。