106台大[illegible]

1. 原CPI = 1 + 0.05×200 + 0.35×0.1×200 = 18

後CPI = 1 + 0.05×200 + 0.35×0.05×200 = 14.5

原cycle time：200ps ⇒ 18×200 = 3600

後cycle time：250ps ⇒ 14.5×250 = 3650

原本架構較好 #

2. code A，可利用平行化架構使計算同時 ⇒ 增加throughput 方式 hide latency

code B 算是序向邏輯的方向

3. (A) 使memory bandwidth增加

(B) 使指令更平衡化

(C) 指令平行化 ⇒ 增Performance.

(D) data locality 不影響效能改善.

4. 選高度data平行可加速方式

(A) DLP

(B) 類似SIMD，另外有control獨立執行緒能力，SIMT ⊇ SIMD

(C) ILP (static)

(D) SMT是TLP，但也有ILP (superscalar)

(E) 也是DLP + ILP

5. 32 reg = 2^5

128 reg = 2^7 ⇒ 每個有用到reg之指令部份多2bit，32+6 = 38 #

6.

(a) Machine 1 = (1457 + 539 + …… + 234) / 12 = 411.29

Machine 2：(1424 + 622 + …… + 191) / 12 = 358.83

Machine 2 #　　$\frac{411}{388}$ #

(b) 429 mcf，利用compiler做最佳化減少CPI

① 共同子問題

② 無意義code消除

③ loop最佳化

7.
(a) CUE cache利用內建memory，on chip cache是SRAM(在process內)

(b) 4K解析通常是 ① 3840×2160 ② 4096×2160 } 傳輸以word為單位 3840×2160×4B = 33.17MB 4096×2160×4B = 35.39MB }.

(c) 要一次讀一堆data下來，若用sequance prefetch可較快處理下一個block fetch

(d) CUE的spatial locality比temporal locality重要，用directed mapping
(主要是hit time成本低而已).

(e) random algorithm ⇒ 因為media前面的page皆不重要，任意置換皆可，且因random不用記錄額外事件loading較少

(f) 檔名當tag <filename, frame>.

8
① if... then else then beq
② switch case① case② ... beq
③ goto j
④ call sub function jr.

(b) 最簡單猜：無條件直接猜跳
最難猜：有條件間接猜跳

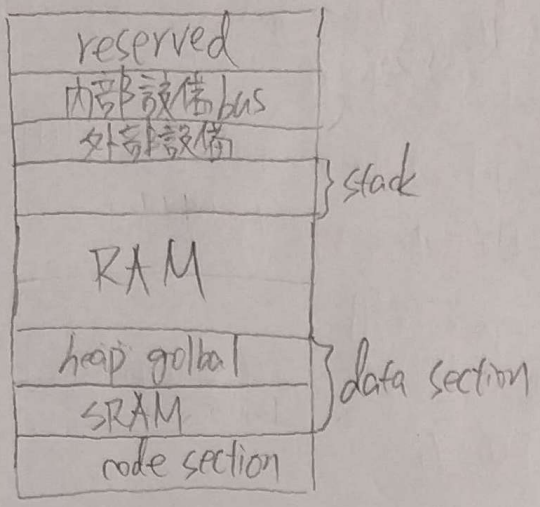
(c) 含有多型的function call ⇒ 用switch選型態適合的，call function & return.

9.
(a) virtual memory，利用secondary store當模擬的memory空間
利用logical address確認memory上是否有page，若無則page fault將page調用.

(b) 在RAM space上找到合適的空間放入segment (Demand segmentation)

9.

(b)



(c) 不能，會造成 thrashing，Page 可載入至 memory，但無法完整載入 cache，所以會造成無法執行的情況

10.

(a) 因 signal(1) 先設好存取錯誤要跳到 catch，18 行產生 access error，所以 PC 被設為 25、26、27

(b) non-preemtive kernel：執行 sys call 時就 disable interrupt 或拒收直到結束

preemtive kernel：允許 sys call 時暫停去處理 interrupt，但插隊中 sys call 處理完若又有新 sys call 會有一致性問題要處理

同步中斷：unmask interrupt：cpu 會立即處理 => 跟 hardware 有關的 interrupt

非同步中斷：mask interrupt：忽略或延遲 => 跟軟體有關的 interrupt

11.

```
1st
wait(mutex);
  measure traffic flow;
  signal(S);
sign(mutex);
```

```
2st
wait(S);
  wait(mutex);
  change the light
  signal mutex)
```

S initial = 0;

12. 若用 hardware 方法實做可確保 atomic，且可做在硬體上出錯率較小；若用 software 方法則較容易制定靈活的方法，但也需控制好同步機制和預防 Deadlock 問題

13.

defs：用檔案方法管理device，管理/dev下設備，每個檔案对應一個設備，不論是否真實存在，建立各自設備檔案，在kernel間交互操作

sysfs：一樣用檔案系統管理，掛載在/sys目錄下，但user空間上也能利用資訊交互操作，可反應目前設備上所有的狀態，只有唯一对應目錄。

udev在user space使用，defs在kernel space使用

Device tree：把硬体資源架構抽出，使整体架構簡單化，不用重新編譯即可更改組態設定，对DTS檔更改即可做小更動

14、共識演算法？