※ 注意：請用 2B 鉛筆作答於答案卡，並先詳閱答案卡上之「畫記說明」。

每題五分。所有題目均為複選題，可以有零個，一個，或一個以上答案。

1. 假設我們有一個序列執行程式，其百分之六十的執行時間，可以切個成四個相同的 thread 來執行。其餘百分之四十的執行時間，則完全無法平行化。請問，根據 Amdahl's law，這個程式經過轉換成四個 thread 的平行化程式，並在四核心的 CPU 上執行時，速度可以是原始序列執行程式的幾倍？
（A）4　（B）2.5　（C）15/8　（D）13/8　（E）20/11

2. 假設我們有一個 CPU，執行一個 ALU 指令，要一個 cycle，一個 load 指令要三個 cycle，一個 branch 指令要五個 cycle。我們現在有一個程式 A，與甲、乙、丙三個 compiler，分別將 A 轉換成 $A_{甲}$、$A_{乙}$、$A_{丙}$三份機器碼（object code）。這三份機器碼執行時，分別會使用下列的指令數：

	ALU 指令數	Load 指令數	Branch 指令數
$A_{甲}$	80 萬	40 萬	50 萬
$A_{乙}$	90 萬	35 萬	48 萬
$A_{丙}$	40 萬	44 萬	58 萬

請問這三份機器碼執行的速度，由快到慢的順序是？
（A）$A_{甲}A_{乙}A_{丙}$（B）$A_{丙}A_{乙}A_{甲}$（C）$A_{甲}A_{丙}A_{乙}$（D）$A_{丙}A_{甲}A_{乙}$（E）$A_{乙}A_{甲}A_{丙}$

3. 請問下列關於 Cache 記憶體的技術與理論，下列哪些敘述是正確的？（複選）
（A） Cache 記憶體能夠發揮效能，是因為 locality of references。
（B） Cache 記憶體能夠發揮效能，是因為 Moore's law。
（C） Direct mapped cache 可以採用 best-fit 演算法，讓一個 block 在 run-time 放到最佳的 cache line 位置。
（D） 在 4-way set associate cache 中，每個 cache block 只能放在固定四個 memory block 中的一個。
（E） 在 miss ratio 是 0.1%時，在使用讀寫速度是記憶體十倍的 Cache 後，讀寫速度可以是原來的九倍以上。

4. 關於 Cache management，下列敘述何者為真？
（A） read-through policy 在 read miss 時，沒有提供任何好處。
（B） write-through policy 在 read operation 時，有可能會產生記憶體的 write operation。
（C） writie-back policy 在 cache block 被 replace 時，不需要將此 cache block 寫回記憶體。
（D） 在 write-miss 發生時，write-allocate policy 不會把要寫入的 cache block 放入記憶體。
（E） Write-through policy 通常不會與 write-allocate policy 搭配使用。

5. 假設我們有一個電腦系統，使用 write-allocate cache。每個 cache block 有四個 words。CPU 每秒鐘會送出 10^7 個記憶體位址，其中 20%是 write operation。而 bus 也可以支援每秒 10^7

見背面

個 word 傳輸。假設在任何瞬間，30%的 cache blocks 都是 dirty。請問下列敘述，何者為真？

（A） 在採用 write-through policy 時，若 hit ratio（read 與 write 合計）是 0.95，則 bus 的 bandwidth 只使用了 25%。

（B） 在採用 write-through policy 時，若 hit ratio（read 與 write 合計）是 0.90，則 bus 的 bandwidth 只使用了 45%。

（C） 在採用 write-back policy 時，若 hit ratio（read 與 write 合計）是 0.95，則 bus 的 bandwidth 只使用了 26%。

（D） 在採用 write-back policy 時，若 hit ratio（read 與 write 合計）是 0.9，則 bus 的 bandwidth 只使用了 52%。

（E） 在採用 write-back policy 時，若 hit ratio（read 與 write 合計）是 0.88，則 bus 的 bandwidth 只使用了 72%。

6. 為了發揮 pipeline 的效率，compiler 在產生機器碼（object code、machine code）時，可以把 for-loop 展開（unrolling）。請問下列敘述，何者為真？

（A） 展開後，可以避免在這個 loop 的機器碼尾端的 branch 指令造成的 pipeline stall。

（B） Compiler 會先把這個 loop 的機器碼產生出來，然後依據 loop 反覆次數，然後將該段機器碼，依序複寫數次，完成 unrolling 步驟。

（C） 這樣的展開，即可以獲得程式執行效率的提昇。

（D） 在 loop 中的資料變數（非 loop-maintenance 變數）存在相依性（dependency），則 compiler 無法將此 loop 展開。

（E） 為了進一步提昇執行效率，compiler 也可能變動展開後的機器碼中指令執行的次序。

7. 為了發揮程式的機器碼在多核心（平行）計算機上的執行效率，compiler 可以將同一個 for-loop 的不同次的 loop-body 執行，配置到不同的核心上執行，達成 thread-level 的 parallelism。譬如下列程式：

```
for (i=0; i<100; i++) a[i] = a[i] + 100;
```

- 對 i∈[0,24]的 “a[i]=a[i]+100;”，可以放到第一個核心上序列執行；
- 對 i∈[25,49]的 “a[i]=a[i]+100;”，可以放到第二個核心上序列執行；
- 對 i∈[50,74]的 “a[i]=a[i]+100;”，可以放到第三個核心上序列執行；
- 對 i∈[75,99]的 “a[i]=a[i]+100;”，可以放到第四個核心上序列執行；

這樣，四個核心同時執行，就可以提升執行速度到四倍。請問在此情況下，下列敘述何者為真？

（A） 下列 C 程式片段的機器碼可以未經 compiler 改寫，直接配置到四個核心上平行執行。

```
for (i=0; i<100; i++) a[i] = a[i] + *x;
```

（B） 下列 C 程式片段的機器碼可以未經 compiler 改寫，直接配置到四個核心上平行執行。

```
for (i=1; i<100; i++) a[i] = a[i] + a[0];
```

（C） 下列 C 程式片段的機器碼可以未經 compiler 改寫，直接配置到四個核心上平行執行。

```
for (i=0; i<100; i++) a[i] = a[i] + b[i];
```

接次頁

（D） 下列C程式片段的機器碼可以未經 compiler 改寫，直接配置到四個核心上平行執行。

```
for (i=0; i<100; i++) { a[i] = a[i]+a[i+100]; a[i+101]=a[i+201]; }
```

（E） 下列 C 程式片段的機器碼：

```
for (i=0; i<100; i++) { a[i] = a[i]+a[i+100]; a[i+101]=a[i+201]; }
```

可以經過 compiler 改寫成下列程式碼後，其 for-loop 可以直接配置到四個核心上平行執行。

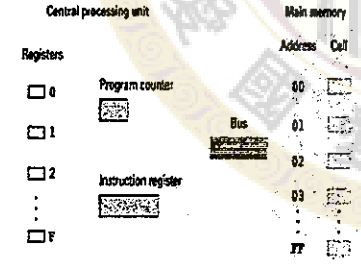
```
a[0]=a[0]+a[100];
for (i=0; i<99; i++) { a[i+101]=a[i+201]; a[i+1] = a[i+1]+a[i+101]; }
a[200]=a[300];
```

8. 在 IEEE 754 的 single-precision floating point 表示法中，是用 32 個 bits 來表達一個浮點數。most significant bit 是 sign bit，接下來八個 bits 是 biased 的 exponent，而最後的 23 個 bits 代表分數部分。請問在此表示法下，下列敘述何者為真？

（A）00111111 11000000 00000000 00000000 代表 0.5
（B）01000000 01010000 00000000 00000000 代表 3.25
（C）10111111 01000000 00000000 00000000 代表 -0.25
（D）10111110 10010000 00000000 00000000 代表 -0.28125
（E）01000000 01101000 00000000 00000000 代表 1.625

9. 假設我們有下列的電腦，與其指令組（instruction set）。

8-bit bus and 8-bit registers



12 machine instructions

Op-code	Operand	Description
1	RXY	LOAD reg. R from cell XY.
2	RXY	LOAD reg. R with XY.
3	RXY	STORE reg. R at XY.
4	ORS	MOVE reg. R to S.
5	RST	ADD reg. S and T into R. (2's comp.)
6	RST	ADD reg. S and T into R. (floating pt.)
7	RST	OR reg. S and T into R.
8	RST	AND reg. S and T into R.
9	RST	XOR reg. S and T into R.
A	ROX	ROTATE reg. R X times.
B	RXY	JUMP to XY if reg. R = reg. 0.
C	000	HALT.

假設記憶體內有下列內容，而且 program counter 的內容是 0。請問這個電腦程式執行結束後，下列敘述何者為真？

記憶體位址（十六進位）	記憶體內容（十六進位）
00	24
01	08
02	13
03	12
04	20
05	00

見背面

06	21
07	FE
08	53
09	34
0A	54
0B	41
0C	B4
0D	10
0E	B0
0F	08
10	C0
11	00
12	1A

（A）reg. 0 的內容為 3（十進位） （B）reg. 1 的內容為-2（十進位） （C）reg. 2 的內容為 0 （D）reg. 3 的內容為 46（十進位） （E）reg. 4 的內容為 0

10. 請問在區分 ISA（Instruction Set Architecture）的類別時，下列哪些敘述是正確的？
（A） register-memory ISA 對 register 與記憶體（memory）的讀寫，採取嚴格分割。因此除了，load 與 store 只另外，其他指令不能直接讀寫記憶體中的資料變數。
（B） 80x86 是 RISC 結構。
（C） 所有的 MIPS 指令都是相同長度。
（D） 80x86 因為是採取 RISC 結構，因此他的 branch instruction，如 BE、BNE，都只能檢驗 register 的內容，作為 branch 與否的依據。
（E） MIPS 讀寫的物件位址需要是 aligned。

11. 在作業系統的運作中，保障下列兩項目標，是極端緊要之事。
- 保護系統資源不被 user app 濫用，
- 防止 user apps 之間的不當干擾

但在同時，作業系統也需要保證他比一般 user app，能執行更多的硬體功能，以完成全電腦系統運作。在這方面，下列哪些敘述為真？
（A） 電腦硬體需具有 user mode 與 kernel mode（supervisor mode）兩種操作模式。
（B） 限制 user app 對某些硬體狀態變數，只能讀，不能寫。
（C） 利用 system call 指令，讓電腦可以由 kernel mode 轉換成 user mode。
（D） 在某些硬體事件發生時，可以強制將 CPU 執行權，由 user app 轉換到作業系統。
（E） 利用硬體機制，限制 user app 可以讀寫的記憶體位址範圍。

12. 請問在作業系統中 process 的管理上，下列哪些敘述為真？
（A） context switch 是發生在兩個 user apps 之間，對 processor 執行權的轉換。
（B） 對一個處於 running 狀態的 process，在其 PCB 中，program counter 欄位的值，是他

接次頁

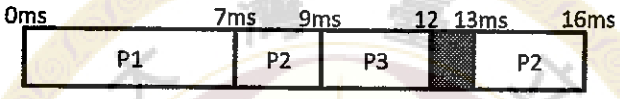
目前正在執行的指令的記憶體位址。

（C） 在等待事件完成，才能恢復執行的 process，其 PCB 都放在 waiting queue 中。

（D） 對 process 執行狀態的管理，作業系統使用兩個 queues，一個是 ready queue，一個是 waiting queue。

（E） Long-term scheduler 的目的，是調整系統的 multiprogramming level。

13. 假設我們有 P1、P2、P3 等三個 process 在 0ms 時，同時進入 ready 狀態。經過 CPU scheduler 安排，立刻產生下列 schedule，將三個 process 都執行完畢。在 12ms 到 13ms 之間，由於 P2 在做 I/O，因此沒能使用 CPU。



請問下列敘述，何者為真？

（A）Average response time 是 16/3

（B）Average waiting time 是 16/3

（C）Average turnaround time 是 35/3

（D）Average throughput 是 16/3

（E）CPU utilization factor 是 15/16

14. Dekker 的 mutual exclusion 演算法，是目前已知，第一個能夠對兩個 thread 解決 critical section 問題的演算法。這個演算法，對 process Pk，k∈{0,1}，其 pseudo-code 如下。

```
Pk:
   flag[k] = true;
   while (CONDITION_X) {
      if (turn ≠ k) {
         flag[k] = false;
         while (turn ≠ k) {
           // busy wait
         }
         flag[k] = true;
   } }

   // critical section
   ...
   turn    = 1-k;
   flag[k] = false;
   // remainder section
```

請問關於這演算法，下列敘述何者為真？

（A）*CONDITION_X* 應該是 flag[1-k] == true

（B）*CONDITION_X* 應該是 flag[1-k] == false

（C）這個演算法要能正確執行，flag[0]與 flag[1]的 initial values 其中一個必須是 false。

（D）這個早期的演算法，並不保證 starvation-freedom，也就是不保證 bounded waiting。

（E）若啟動 compiler 的 out-of-order execution optimization，則此演算法產生的機器碼，不再保證 critical section 的 mutual exclusion。

15. 在 database 的 log-based recovery 技術中，下列哪些敘述為真？

（A） 在 log 中記錄的每筆 transaction 的 operation，必須把運算式記錄下來。

（B） 在 redo 時，要把 crash 時還沒有 committed 的 transactions 全部完成。

（C） 在 undo 時，要把 crash 時已經 committed 的 transactions 都取銷。

（D） 可以利用 checkpoint 的技術，讓每次 recovery 只要從最近一次 checkpoint 開始進行即可。

（E） 使用 log-based recovery 技術，可以確保產生的 transaction schedules 都是 serializable。

16. 在使用 deadlock avoidance 的 Banker's 演算法時，如果我們有三種資源 A、B、C，分別有 10、5、7 個 copies。現在有一個狀態，我們把 A、B、C 三種資源，配置給 P0、P1、P2 三個 processes，並以下列 Allocation 與 Max 矩陣，描述資源配置的情形。

Allocation	A	B	C
P0	0	1	2
P1	4	0	1
P2	3	1	2

Max	A	B	C
P0	9	5	8
P1	7	2	2
P2	9	5	2

請問下列敘述何者為真？

（A）這不是一個 safe 狀態。

（B）P0 P1 P2 是一個 safe sequence。

（C）P2 P0 P1 是一個 safe sequence。

（D）P1 P0 P2 是一個 safe sequence。

（E）P1 P2 P0 是一個 safe sequence。

17. 在 virtual memory 中，在執行下列 page reference string 後，

0, 1, 2, 3, 4, 1, 5, 0, 2, 1, 3, 2, 4, 1, 5, 2, 0, 1, 3, 2, 4, 1, 5

下列敘述何者為真？

接次頁

（A） 假設一個 process 有四個 frames，在 LRU 的 page replacement 演算法下，四個 frames 裡的 page numbers 分別是 1、2、4、5。
（B） 假設一個 process 有四個 frames，在 FIFO 的 page replacement 演算法下，四個 frames 裡的 page numbers 分別是 1、3、4、5。
（C） 假設一個 process 有四個 frames，在 second chance 的 page replacement 演算法下，四個 frames 裡的 page numbers 分別是 1、3、4、5。（假設演算法每次都從第一個 frame 開始算。而且每次一個新的 page 被放進記憶體，其 reference bit 會立刻被設為 1。）
（D） 假設一個 process 有三個 frames，在 LRU 的 page replacement 演算法下，三個 frames 裡的 page 分別是 1、4、5。
（E） 假設一個 process 有三個 frames，在 FIFO 的 page replacement 演算法下，三個 frames 裡的 page 分別是 1、2、5。

18. 在作業系統的 paging memory 中，下列敘述何者為真？
（A） TLB 是發揮 paging memory 效能的關鍵軟體技術。
（B） TLB 能夠有效的原因，是 locality of references。
（C） TLB 是一種 associative memory。
（D） 在 paging memory 中，不會有 external fragmentation 的問題。
（E） 在 paging memory 中，average internal fragmentation 是半個 page size。所以技術上的趨勢，是採用越來越小的 page size。

19. 在討論作業系統的 thread management 時，請問下列何者為真？
（A） 可以用 copy-on-write 技術，來降低執行程式，可能需要的 page 數目。
（B） Race condition 是一個重要的作業系統技術，可增加每個 thread 的執行速度。
（C） Thread creation 所需的成本，低於 process creation 的成本。
（D） Thread 之間的資源/資訊共享，可以比 process 之間更有彈性與效率。
（E） Thread pool 的技術，可以降低 thread creation 所需的成本。

20. 假設我們有一個 hard disk，共有 100 個 tracks，編號從 0 到 99。假設我們的磁頭現在 track 50，並且往 track 99 的方向移動。如果此時，我們同時收到在下列 tracks 的讀寫要求。

18、9、48、79、89、60、77、23

請問，下列敘述何者為真？
（A）在採用 SSTF disk scheduling 演算法時，磁頭的移動距離將是 123.
（B）在採用 SCAN disk scheduling 演算法時，磁頭移動的距離將是 139
（C）在採用 C-SCAN disk scheduling 演算法時，磁頭移動的距離將是 196
（D）在採用 LOOK disk scheduling 演算法時，磁頭移動的距離將是 119
（E）在採用 C-LOOK disk scheduling 演算法時，磁頭移動的距離將是 158.

試題隨卷繳回