【例一】在Compiler之lexical phase 有那三種task需要完成？且需要牽涉那些data base？

【解】（一）在Complier 之lexical phase有下列三種task需要完成：

1. 將原始程式分割成一系列的語言基本元素或稱字彙（token）。
2. 建立文字表（literal table）及識別字表（identifier table）
3. 建立齊一符號表（uniform symbol table）

（二）在compiler之Lexical phase需牽涉下列的data base：

1. 原始程式（Source program）——輸入。
2. 終端符號表（Terminal table）——參考（reference）。
3. 文字表（Literal table）——輸出。
4. 識別字表（Identifier table）——輸出。
5. 齊一符號表（Uniform symbol table）——輸出。

【例二】多重選擇題：

The tasks of the lexical analysis phase of a compiler are: (a) to parse the source program into the basic elements of the language (b) to build a literal table and an identifier table (c) to build a uniform symbol table (d) to interpret the reductions.

【解】（a）（b）（c）

【例三】在一編譯程式（Compiler）中，用何法來作字彙的分析（Lexical Analysis）？即如何辨認原始程式（Source Program）中的字彙（Tokens）？是簡述之。

【解】（一）在一編輯程式中，字彙分析是以終端符號表（terminal table）的某些符號（例如：空白、運算符號（operator）及特殊符號等。）為分隔字元（break characters），將原始程式分割成一系列的字彙（tokens），辨認字彙的原則如下：

1. 連續的分隔字元被累積成一個字彙，亦即兩個相鄰的分隔字元之間的字串為一個字彙。
2. 非空白的分隔字元為一個字彙。

（二）字彙分析能辨認終端符號（terminal symbols）、辨別字（identifiers）及文字（literal）等三種類型的字彙，作法如下：

1. 首先將字彙與終端符號表中的項目作比較，若找到與該字彙相同的項目則該字彙是終端符號。
2. 若不滿足1. 則檢查該字彙是否為程式中的變數或暫時儲存位置；若答案肯定，則該字彙屬於識別字。
3. 若不滿足1.及2. 則檢查該字彙是否為常數（constant），若答案肯定，則該字彙屬於文字。
4. 若1.、2.及3.都不滿足則檢查出該字彙有誤並且作錯誤的註記。

【例三】舉出編譯器中四種與機器無關之最佳化（Machine Independent Optimization）及其執行方法。

【解】原始程式經過語彙分析（lexical analysis）、語法分析（syntax analysis）及解釋（interpretation）等階段的編譯程序後，產生一中間矩陣，該矩陣再經過“與機器無關的最佳化”後，產生“與機器無關的最佳化矩陣”；舉出編譯器中四種與機器無關之最佳化及其執行方法如下：

（一）刪除共同的副式子（common subexpression）；執行方法如下：

1. 將中間矩陣的項目（entry）（內容包過運算符號、兩個運算元以及分別指向前面一個項目和後面一個項目的兩個指標等）以特定的方式放置，以便能辨認出共同的副式子；例如：將運算符號是+或*之矩陣項目的兩個運算元按照字母順序排列。
2. 辨認相同的副式子（亦即矩陣的項目中運算符號及兩個運算元都相同者）。
3. 將相同的副式子其中一個保留，其餘刪除。
4. 將中間矩陣中所有參考到被刪除項目的相關內容修正。
5. 反覆執行步驟1.至4.，直到中間矩陣沒有共同副式子為止。

（二）在編譯時計算（compile time compute），亦即在編譯時將常數（constant）間的運算先予計算；執行方法如下：

1. 找尋中間矩陣的項目中內容包含運算符號而且兩個運算元都是常數者。

2. 當找到滿足1. 的項目時，首先計算此項目並且產生新的常數，再將這個項目刪除，然後將中間矩陣中所有參考到此項目的運算元用產生新的常數取代之以及修正中間矩陣中其他與此項目相關的內容。

3. 反覆執行1.及2.，直到中間矩陣的項目中無兩個運算元都常數者為止。

（三）布林式子的最佳化（Boolean expression optimization）；執行方法是利用布林式子的特性來縮短計算。例如：一指述IF L1 OR L2 OR L3 THEN F1其中L1，L2及L3均為邏輯式子；F1是由一個或多個指述所組成；將採用布林代數的功能與否分別說明如下：

1. 不採用布林代數的功能時：L1，L2及L3全部都要計算及測試計算結果是否為真，當三者中任一為真時才執行F1。

2. 採用布林代數的功能時：首先計算及測試L1，當L1為真則L2及L3都不必計算及測試便執行F1，否則才計算及測試L3，當L3為真時，才執行F1；如此這般便可縮短計算。

（四）將迴路（loop）中不變的計算式子移出迴路外；執行方法如下：

1. 辨認不變的計算式子。
2. 找出放置不變的計算式子之適當位置——通常是迴路起始的前一個位置。
3. 移動不變的計算式子——只需更新小部分之向前及向後的指標即可。

【例六】寫出下列程式經過Compile的語意分析階段所建立之中間碼，如再經與機器無關之最佳化則所得結果（中間碼）？

.

.

.

A=2+3*(A+B)-(A+B+C)*D

X=(A+B)*(2+3)

.

.

.

【解】（一）算術式A=2+3*(A+B)-(A+B+C)*D經語意分析所建立成的中間碼以及再經與機器無關之最佳化產生的中間碼如下：

序號	矩陣	中間碼	與機器無關最佳化矩陣	與機器無關之最佳化中間碼
1	+ A B	L 1,A A 1,B ST 1,M1	+ A B	L 1,A A 1,B ST 1,M1
2	* 3 M1	L 1,=F`3` M 0,M1 ST 1,M3	* 3 M1	L 1,=F`3` M 0,M1 ST 1,M2
3	+ 2 M2	L 1,=F`2` A 1,M2 ST 1,M3	+ 2 M2	L 1,=F`2` A 1,M2 ST 1,M3
4	+ A B	L 1,A A 1,B ST 1,M4		

5	+ M4 C	L 1,M4 A 1,C ST 1,M5	+ M1 C	L 1,M1 A 1,C ST 1,M5
6	* M5 D	L 1,M5 M 0,D ST 1,M6	* M5 D	L 1,M5 M 0,D ST 1,M6
7	- M3 M6	L 1,M3 S 1,M6 ST 1,M7	- M3 M6	L 1,M3 S 1,M6 St 1,M7
8	= A M7	L 1,M7 ST 1,A	=A M7	L 1,M7 ST 1,A

註：此處刪除共同的副式子（common subexpression）“A+B”，以達成與機器無關的最佳化。

（二）算術式X＝（A＋B）＊（2＋3）經語意分析所建立成的中間碼以及再經過與機器無關之最佳化產生的中間碼如下：

序號	矩陣	中間碼	與機器無關 最佳化矩陣	與機器無關之 最佳化中間碼
1	+ A B	L 1,A A 1,B ST 1,M1	+ A B	L 1,A A 1,B ST 1,M1
2	+ 2 3	L 1,=F`2` A 1,=F`3` ST 1,M2		
3	* M1 M2	L 1,M1 M 0,M2 ST 1,M3	* M1 5	L 1,M1 M 0,=F`5` ST 1,M3

序號	矩陣	中間碼	與機器無關 最佳化矩陣	與機器無關之 最佳化中間碼
4	= X M3	L 1,M3 ST 1,X	= X M3	L 1,M3 ST 1,X

註：此處將常數2+3先行計算，以達成與機器無關的最佳化。

A=2+3*(A+B)-(A+B+C)*D

X=(A+B)*(2+3)

【例七】Give three examples of machine-dependent optimization.

【解】舉出三個“與機器有關之最佳化”的例子如下：

一 刪除多餘的load（L）及store(ST)的指令。

二 執行最佳化時儘量利用尚未使用到的暫存器（register）

三 利用執行速度較快的指令取代執行速度較慢的指令；例如：以執行速度較快的RR態指令取代執行速度較慢的RX態指令。

註：舉一實例綜合說明之，已知一算術式

A＝B＊C＋D＋E＊F，該算術式對應的矩陣及最佳化前數碼如下：

矩陣	最佳化前之數碼
(1) ＊ B C	(1) L 1 ,B
(2) ＋ M1 D	(2) M 0 ,C
(3) ＊ E F	(3) ST 1 ,M1
(4) ＋ M2 M3	(4) L 1 ,M1
(5) ＝ A A4	(5) A 1 ,D
	(6) ST 1 ,M2
	(7) L 1 ,E
	(8) M 0 ,F
	(9) ST 1 ,M3

矩陣	最佳化前之數碼
	(10) L 1 ,M2
	(11) A 1 ,M3
	(12) ST 1 ,M4
	(13) L 1 ,M4
	(14) ST 1 ,A

接上圖

茲依前面的例子，將上述數碼執行“與機器有關之最佳化”，說明如下：

(一)因為一個已存放在暫存器中之數值如果立刻要使用到，則該數值不需要儲存；而若一個數值已經存放在暫存器中，則不必再重新載入該數值；所以將上述數碼刪除多餘的load及store指令後，結果如下：

(1) L 1 ,B
(2) M 0 ,C
--
--
(3) A 1 ,D
(4) ST 1 ,M2
(5) L 1 ,E
(6) M 0 ,F
(7) ST 1 ,M3
(8) L 1 ,M2
(9) A 1 ,M3
--
--
(10)ST 1 ,A

（二）利用尚未使用到的暫存器以及執行速度較快的RR態指令，將上述數碼再作最佳化後，結果如下：

(1) L 1 ,B
(2) M 0 ,C
(3) A 1 ,D
(4) L 3 ,E
(5) M 2 ,F
(6) AR 1 ,3
(7) ST 1 ,A