1. "For" loop conversion with *cmp*, *jnz*

```
public class Processor {

   //eax, ebx, ecx, edx, and flags have 32 bits, just like an int in java
   public static int eax;
   public static int ebx;
   public static int ecx;
   public static int edx;

   public static void main(String[] args) {
      ebx = 5;

      for(eax = 0; eax < ebx; eax++) {
         ecx++;
      }
      //value of ecx should be 5 after the loop terminates.
   }
}
```

```
.code
main proc
      move eax, 0     ;Initialization
      move ebx, 5     ;Initialization
      move ecx, 0      ;Initialization
      ForLoop:
            inc ecx
            inc eax
            cmp eax, ebx
      jnz ForLoop
     invoke ExitProcess, 0
   main endp
end main
```

2. if…else if……else……conversion with *jmp, jl, je*

```
if( eax > 0) {
    ecx = 1;
}
else if (eax < 0) {
    ecx = 2;
}
else {
    ecx = 3;
}
```

```
.code;
        main proc
                mov eax, 1              ;Initialization
                cmp eax, 0              ;compare eax to 0

                jl eaxElseIf            ;if eax < 0 jump to eaxElseIf
                je eaxElse              ;if eax == 0 jump to eaxElse
                mov ecx, 1              ;if eax > 0 don't jump
                jmp ifEnd               ;jump to ifEnd
                eaxElseIf:
                        mov ecx, 2      ;set ecx to 2
                        jmp ifEnd       ;jump to ifEnd
                eaxElse:
                        mov ecx, 3      ;set ecx to 3
                ifEnd:
                     ………….

                invoke ExitProcess, 0
        main endp
end main
```

3. Short-circuit conversion with *cmp*, *jbe*

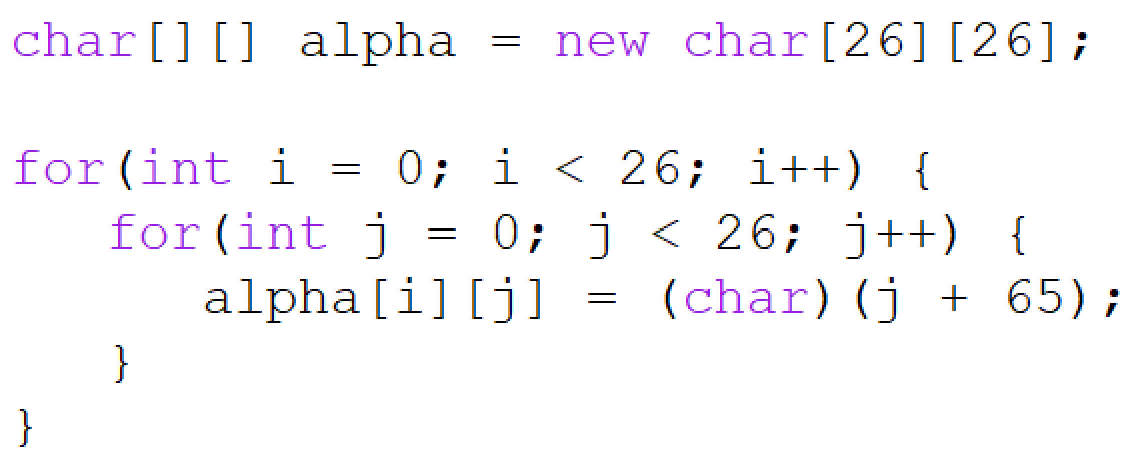
```
if (eax > 0 && ebx > 0)
      ecx = 4;


.code;
        main proc
                mov eax, 1          ;Initialization
                mov ebx, 1          ;Initialization

                cmp eax, 0          ;check eax first
                jbe False           ;jump to false if eax > 0 is false
                cmp ebx, 0          ;check ebx second
                jbe False           ;jump to fase if ebx > 0 is false
                mov ecx, 4          ;if both pass set ecx to 4

                False:
                     ................
                invoke ExitProcess, 0
        main endp
end main
```

4. 2D array conversion with *cmp*, *jmp*, *jz*



```
char[][] alpha = new char[26][26];

for(int i = 0; i < 26; i++) {
    for(int j = 0; j < 26; j++) {
        alpha[i][j] = (char)(j + 65);
    }
}
```

Version 1 (nested loop)

```
.data
	alpha byte 26 * 26 dup(0)
.code
	main proc
		mov bl, 65		;"A" in ASCII
		mov ecx, 26		;Number of columns
		mov edi, 0		;row counter
		mov esi, 0		;position
		OuterLoop:
			InnerLoop:
				mov alpha[esi], bl
				inc bl
				inc esi
			loop InnerLoop
		cmp edi, 26
		jz Done
		inc edi
		mov bl, 65
		mov ecx, 26
		jmp OuterLoop

		Done:
		invoke ExitProcess, 0
	main endp
end main
```

Version 2 (single loop)

```
.data
	a byte "ABCDEFGHIJKLMOPQRSTUVWXYZ"
	alpha byte 26 * 26 dup(0)
.code
	main proc
		mov esi, offset a
		mov edi, offset alpha
		mov ebx, 26
		L:
			mov ecx, 25
			rep movsb
			dec ebx
			cmp ebx, 0
		jnz L
		invoke ExitProcess, 0
	main endp
end main
```

5. Struct. Set the value of “lastName” to “Smith” with a given struct Employee.

```
Employee struct
	idNum byte ?
	lastName byte 10 dup(?)
	years byte 0
Employee ends




.data
	worker Employee <>
	lastName byte "Smith"
.code
	main proc
		mov eax, sizeof worker
		mov worker.idnum, 50h
		mov worker.years, 5
		mov esi, offset lastName
		mov edi, offset worker.lastName
		mov ecx, lengthof lastName
		rep movsb
		;mov worker.lastName, "Smith"
		invoke ExitProcess, 0
	main endp
end main
```