	Silesian University of Technology **Department of Graphics, Computer Vision and Digital Systems**			
Year	**Type*: SSI/NSI/NSM**	**Subject: Assembler Programming Languages**	**Group**	**Section**
2024/2025	SSI	**APL – LAB**	**1**	**1**
Tutor:	dr inż. Adam Opara		**Class date: (week day, hour)**	
Section:	**1. Piotr Copek** **2. Zuzanna Micorek**		**06/06/2025**	
Contact Email:	**pc21339@student.polsl.pl**		**8.30 – 10.00**	

Report

Task 1

Create procedure `CheckSSECpp` returning 1 if procesor supports `SSE`.

```
extern "C" int CheckSSECpp() {
    int cpuInfo[4];
    __cpuid(cpuInfo, 1);
    return (cpuInfo[3] & (1 << 25)) ? 1 : 0;
}
```

`CheckSSECpp` uses the `C++` intrinsic `__cpuid()` to check if the processor supports `SSE` (Streaming SIMD Extensions). When `__cpuid` is called with `EAX=1`, bit 25 of `EDX` in the returned `cpuInfo` array indicates whether `SSE` is available. The function returns 1 if `SSE` is supported, otherwise 0.

Task 2

Create an assembly `DLL` with a procedure `CheckSSEAsm` checking availability of `SSE` instructions, and with `RadToDegAsm` function.

```
.code
PUBLIC CheckSSEAsm
PUBLIC RadToDegAsm

CheckSSEAsm PROC
    mov     eax, 1
    cpuid
    bt      edx, 25           ; check SSE bit (bit 25 in EDX)
    setc    al                ; set AL = 1 if bit was set
    movzx   eax, al           ; zero-extend to full EAX
    ret
CheckSSEAsm ENDP
```

`CheckSSEAsm` is a pure x64 assembly implementation that uses the `cpuid` instruction to query processor features. It sets `EAX` to 1 and executes `cpuid`, then checks bit 25 of `EDX` using `bt`. If the bit is set (`SSE` supported), the result is 1, otherwise 0.

```
RadToDegAsm PROC
    movsd xmm1, qword ptr [dbl180]
    movsd xmm2, qword ptr [dblPI]
    divsd xmm1, xmm2
    mulsd xmm0, xmm1
    movsd qword ptr [tempDeg], xmm0
    cvttsd2si rax, xmm0
    cvtsi2sd xmm3, rax
    subsd xmm0, xmm3
    movsd xmm4, qword ptr [dbl60]
    mulsd xmm0, xmm4
    imul rax, 100
    cvtsi2sd xmm3, rax
    addsd xmm0, xmm3
    ret
RadToDegAsm ENDP

.data
dblPI  REAL8 3.141592653589793
dbl180 REAL8 180.0
dbl60  REAL8 60.0
tempDeg REAL8 0.0

END
```

`RadToDegAsm` is a SIMD-based function written in x64 assembly using `SSE2` (`xmm` registers). It converts an angle in radians to the `NMEA` coordinate format (DDDMM.mmmm), used in GPS.

Procedure:

- multiplies the input radians by $\frac{180}{\pi}$ to get degrees
- splits degrees into integer and fractional parts
- converts the fractional part to minutes
- returns the final result as `intDeg` $\times$ 100 $+$ `minutes`

All calculations use floating-point SIMD registers (`xmm0` - `xmm4`).

Task 3

Create procedure `RadToDegCpp` in `C++`.

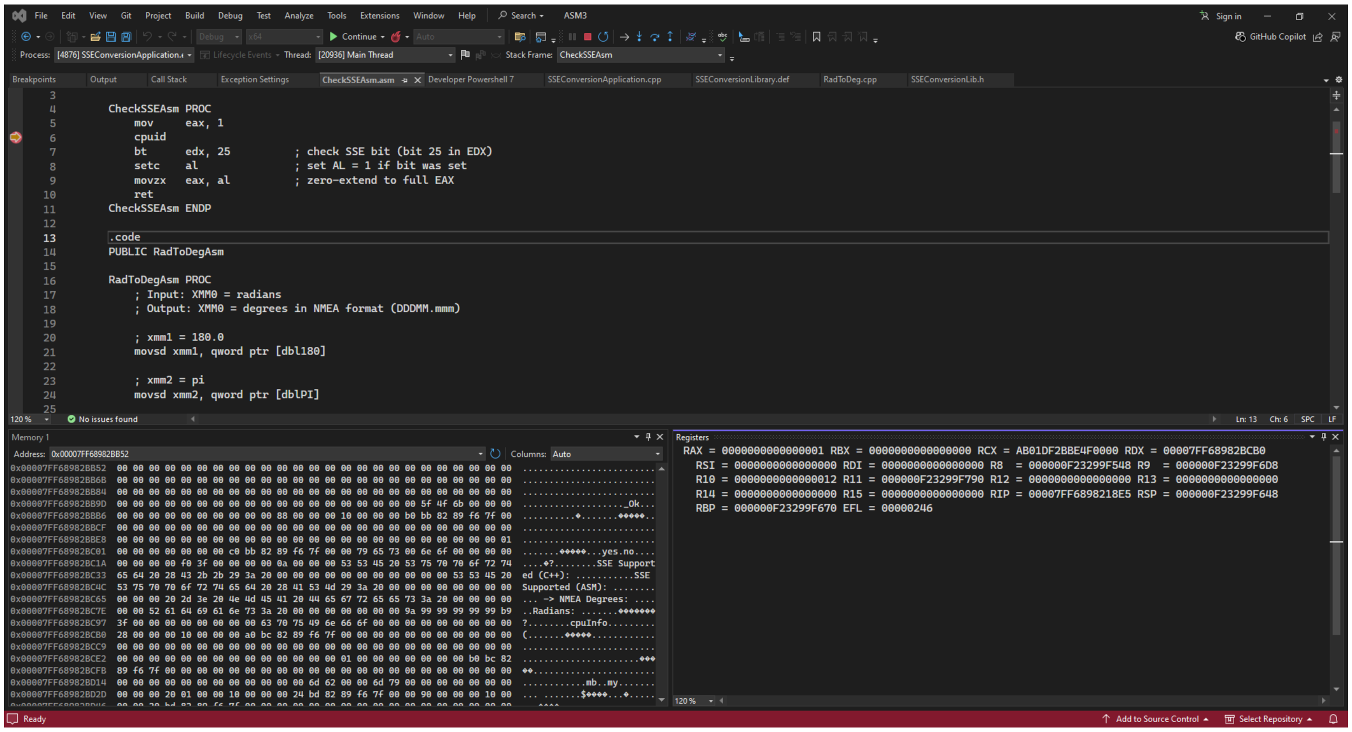
```cpp
extern "C" double RadToDegCpp(double radians, int sseEnabled) {
    double dDeg = 0.0;
    if (sseEnabled) {
        __m128d ma, mb, mDeg, mx, my, mz, mk;
        ma = _mm_set_sd(180.0);
        mb = _mm_set_sd(3.141592653589793);
        mDeg = _mm_div_pd(ma, mb);
        ma = _mm_set_sd(radians);
        mb = _mm_mul_pd(ma, mDeg);
        dDeg = mb.m128d_f64[0];
        int nDeg = (int)dDeg;
        double dRem = dDeg - nDeg;
        ma = _mm_set_sd(100.0);
        mb = _mm_set_sd(60.0);
        mx = _mm_set_sd(dRem);
        mz = _mm_mul_pd(mx, mb);
        mk = _mm_set_sd(nDeg * 100.0);
        my = _mm_add_pd(mk, mz);
        return my.m128d_f64[0];
    } else {
        double dDeg = radians * 180.0 / 3.141592653589793;
        int nDeg = (int)dDeg;
        double dRem = dDeg - nDeg;
        return nDeg * 100.0 + dRem * 60.0;
    }
}
```

`RadToDegCpp` performs the same conversion from radians to `NMEA` degrees format as the assembly version, but in `C++`. It supports two modes:

- `SSE2` mode: uses `_mm_set_sd`, `_mm_div_pd`, `_mm_mul_pd`, and other intrinsics to perform vectorized floating-point math.
- Fallback mode: if `SSE` is not available, the calculation is done using standard scalar operations.

Task 4

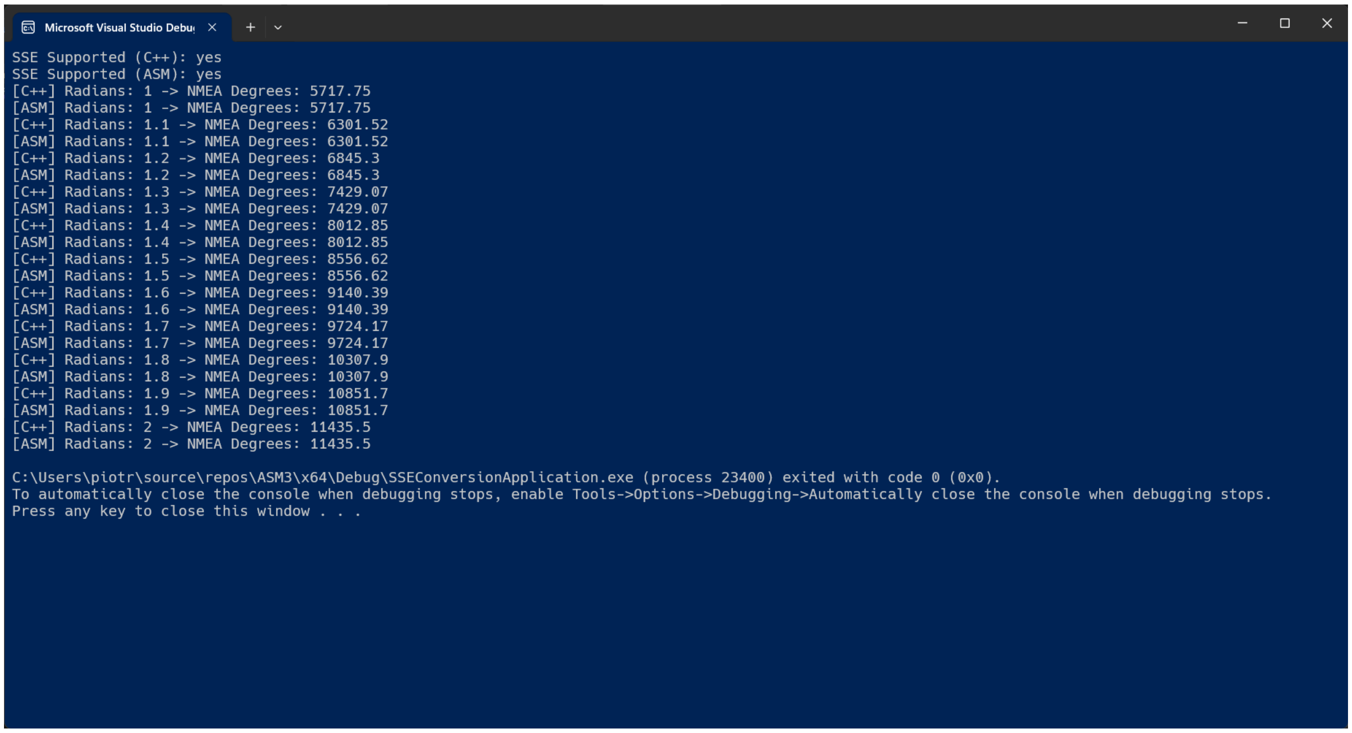
Compile and run an application that calls procedures `CheckSSExxx` and `RadToDegxxx`.



```
CheckSSEAsm PROC
    mov     eax, 1
    cpuid
    bt      edx, 25          ; check SSE bit (bit 25 in EDX)
    setc    al               ; set AL = 1 if bit was set
    movzx   eax, al          ; zero-extend to full EAX
    ret
CheckSSEAsm ENDP

.code
PUBLIC RadToDegAsm

RadToDegAsm PROC
    ; Input: XMM0 = radians
    ; Output: XMM0 = degrees in NMEA format (DDDMM.mmm)

    ; xmm1 = 180.0
    movsd xmm1, qword ptr [dbl180]

    ; xmm2 = pi
    movsd xmm2, qword ptr [dblPI]
```

```
Address: 0x00007FF68982BB52
0x00007FF68982BB52  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................
0x00007FF68982BB6B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................
0x00007FF68982BB84  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................
0x00007FF68982BB9D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 5f 4f 6b 00 00 00  ...................._Ok...
0x00007FF68982BBB6  00 00 00 00 00 00 00 00 00 00 88 00 00 00 10 00 00 00 b0 bb 82 89 f6 7f 00  ..........�.......�����..
0x00007FF68982BBCF  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................
0x00007FF68982BBE8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01  .........................
0x00007FF68982BC01  00 00 00 00 00 00 00 c0 bb 82 89 f6 7f 00 00 79 65 73 00 6e 6f 00 00 00 00  .......�����...yes.no....
0x00007FF68982BC1A  00 00 00 00 f0 3f 00 00 00 00 0a 00 00 00 53 53 45 20 53 75 70 70 6f 72 74  ....�?........SSE Support
0x00007FF68982BC33  65 64 20 28 43 2b 2b 29 3a 20 00 00 00 00 00 00 00 00 00 00 00 53 53 45 20  ed (C++): ...........SSE
0x00007FF68982BC4C  53 75 70 70 6f 72 74 65 64 20 28 41 53 4d 29 3a 20 00 00 00 00 00 00 00 00  Supported (ASM): ........
0x00007FF68982BC65  00 00 00 20 2d 3e 20 4e 4d 45 41 20 44 65 67 72 65 65 73 3a 20 00 00 00 00  ... -> NMEA Degrees: ....
0x00007FF68982BC7E  00 00 52 61 64 69 61 6e 73 3a 20 00 00 00 00 00 00 00 9a 99 99 99 99 99 b9  ..Radians: .......�������
0x00007FF68982BC97  3f 00 00 00 00 00 00 00 00 63 70 75 49 6e 66 6f 00 00 00 00 00 00 00 00 00  ?........cpuInfo.........
0x00007FF68982BCB0  28 00 00 00 10 00 00 00 a0 bc 82 89 f6 7f 00 00 00 00 00 00 00 00 00 00 00  (.......�����............
0x00007FF68982BCC9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................
0x00007FF68982BCE2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 b0 bc 82  ......................���
0x00007FF68982BCFB  89 f6 7f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ��.......................
0x00007FF68982BD14  00 00 00 00 00 00 00 00 00 00 00 00 6d 62 00 00 6d 79 00 00 00 00 00 00 00  ............mb..my.......
0x00007FF68982BD2D  00 00 00 20 01 00 00 10 00 00 00 24 bd 82 89 f6 7f 00 00 90 00 00 00 10 00  ... .......$����...�.....
```

```
RAX = 0000000000000001 RBX = 0000000000000000 RCX = AB01DF2BBE4F0000 RDX = 00007FF68982BCB0
  RSI = 0000000000000000 RDI = 0000000000000000 R8  = 000000F23299F548 R9  = 000000F23299F6D8
  R10 = 0000000000000012 R11 = 000000F23299F790 R12 = 0000000000000000 R13 = 0000000000000000
  R14 = 0000000000000000 R15 = 0000000000000000 RIP = 00007FF6898218E5 RSP = 000000F23299F648
  RBP = 000000F23299F670 EFL = 00000246
```



```
SSE Supported (C++): yes
SSE Supported (ASM): yes
[C++] Radians: 1 -> NMEA Degrees: 5717.75
[ASM] Radians: 1 -> NMEA Degrees: 5717.75
[C++] Radians: 1.1 -> NMEA Degrees: 6301.52
[ASM] Radians: 1.1 -> NMEA Degrees: 6301.52
[C++] Radians: 1.2 -> NMEA Degrees: 6845.3
[ASM] Radians: 1.2 -> NMEA Degrees: 6845.3
[C++] Radians: 1.3 -> NMEA Degrees: 7429.07
[ASM] Radians: 1.3 -> NMEA Degrees: 7429.07
[C++] Radians: 1.4 -> NMEA Degrees: 8012.85
[ASM] Radians: 1.4 -> NMEA Degrees: 8012.85
[C++] Radians: 1.5 -> NMEA Degrees: 8556.62
[ASM] Radians: 1.5 -> NMEA Degrees: 8556.62
[C++] Radians: 1.6 -> NMEA Degrees: 9140.39
[ASM] Radians: 1.6 -> NMEA Degrees: 9140.39
[C++] Radians: 1.7 -> NMEA Degrees: 9724.17
[ASM] Radians: 1.7 -> NMEA Degrees: 9724.17
[C++] Radians: 1.8 -> NMEA Degrees: 10307.9
[ASM] Radians: 1.8 -> NMEA Degrees: 10307.9
[C++] Radians: 1.9 -> NMEA Degrees: 10851.7
[ASM] Radians: 1.9 -> NMEA Degrees: 10851.7
[C++] Radians: 2 -> NMEA Degrees: 11435.5
[ASM] Radians: 2 -> NMEA Degrees: 11435.5

C:\Users\piotr\source\repos\ASM3\x64\Debug\SSEConversionApplication.exe (process 23400) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Conclusions

In this lab, I learned how to detect SSE support using both C++ and x64 Assembly. The SSE detection worked correctly in both languages, and the conversion produced accurate results. Writing SIMD code in Assembly was more complex, but it gave better control over registers. This lab helped me understand how modern CPUs handle floating-point operations efficiently.