Vježba L03

Jakov Spahija

4. lipnja 2021.

Sadržaj

1. DXM Math

Ugrađene definicije funkcije od strane kompajlera *intrinsics*, postoje za SIMD registre. Intrinzički tipovi `_m64`, `_m128`, `_m256` i `_m512` definiraju SIMD tipove za 64-bit XMM, 128-bit MMX, 256-bit YMM te 512-bit ZMM registre, respektivno[1].
`DirectXMath.h` koristi `xmmintrin.h` za intrinzičke tipove.

`__vectorcall` konvencija poziva koja je definirana za SIMD, koja za razliku od zadane `__cdecl` konvencije, učitava veći broj argumenata direktno u dodatne XMM registre, koji su posebni SSE registri[2].

Eksplicitno je zabranjeno korištenje liste argumenata varijabilne dužine, jer kao i kod `__stdcall`, pozvana funkcija nema informaciju o broju argumenata koji su proslijeđeni na stog, pa nije u stanju brisati argumente sa stoga.

`__vectorcall konvencija`

```
inline DirectX::XMVECTOR* XM_CALLCONV
f( DirectX::FXMVECTOR v1, DirectX::FXMVECTOR v2,
      DirectX::FXMVECTOR v3, DirectX::GXMVECTOR v4,
      DirectX::HXMVECTOR v5, DirectX::HXMVECTOR v6,
      DirectX::CXMVECTOR v7, DirectX::CXMVECTOR v8){
```

`XM_CALLCONV` se unutar DirectXMath definira kao `__vectorcall`.

Prvih šest argumenata se mogu proslijediti direktno u registre, koji se prenose po vrijednosti, dok se ostali proslijeđuju po referenci[3]:

`rezolucija argumenata za __vectorcall na x64 platformi`

```
typedef const XMVECTOR FXMVECTOR;
typedef const XMVECTOR GXMVECTOR;
typedef const XMVECTOR HXMVECTOR;
typedef const XMVECTOR& CXMVECTOR;
typedef const XMMATRIX FXMMATRIX;
typedef const XMMATRIX& CXMMATRIX;
```

[1] `https://www.agner.org/optimize/calling_conventions.pdf` *7.2 Passing and returning SMID types*

[2] `https://en.wikipedia.org/wiki/Advanced_Vector_Extensions#Advanced_Vector_Extensions`

[3] `https://docs.microsoft.com/en-us/windows/win32/dxmath/pg-xnamath-internals#calling-conventions`

XMLoad funkcije se koriste da bi se XM tipovima predali podaci varijabli koje su definirani sa intrizičkim tipovima SSE registara. Podaci se prenose preko pokazivača, a XMLoad funkcije samo referenciraju podatke bez kopiranja.

Prijepis 1.1. DirectXMathConvert.inl: SSE XMLoadFloat4x4

```
#elif defined(_XM_SSE_INTRINSICS_)
    XMMATRIX M;
    M.r[0] = _mm_loadu_ps( &pSource->_11 );
    M.r[1] = _mm_loadu_ps( &pSource->_21 );
    M.r[2] = _mm_loadu_ps( &pSource->_31 );
    M.r[3] = _mm_loadu_ps( &pSource->_41 );
    return M;
#endif
```

2. DXM Vectors

Prijepis 2.1. Functions.h

```cpp
#pragma once
#include<iostream>
#include<string>
#include<sstream>
#include<xmmintrin.h>
#include<windows.h>
#include<DirectXMath.h>


inline DirectX::XMVECTOR* XM_CALLCONV
f( DirectX::FXMVECTOR v1, DirectX::FXMVECTOR v2,
        DirectX::FXMVECTOR v3, DirectX::GXMVECTOR v4,
        DirectX::HXMVECTOR v5, DirectX::HXMVECTOR v6,
        DirectX::CXMVECTOR v7, DirectX::CXMVECTOR v8){
        std::cout << typeid(v1).name() << std::endl;
        return new DirectX::XMVECTOR;
}


std::string ToString(DirectX::XMFLOAT4& vec){
        std::stringstream out;
        out << "[" << vec.x << ", " << vec.y << ", " << vec.z << ", " << vec.w << "]";
        return out.str();
}

class C {
public:
        C( DirectX::FXMVECTOR v1, DirectX::FXMVECTOR v2,
                DirectX::FXMVECTOR v3, DirectX::GXMVECTOR v4,
                DirectX::HXMVECTOR v5, DirectX::HXMVECTOR v6,
                DirectX::CXMVECTOR v7, DirectX::CXMVECTOR v8) {
                std::cout << typeid(v1).name() << std::endl;
        }
};
```

- **Notice that XMVector3Length returns a vector. How can you get the length as a scalar value?**

Program.cpp

```cpp
        XMVECTOR u = XMLoadFloat4(&f4);
        cout << "Duljina: " << XMVector3Length(u).m128_f32[0] << endl;
```

Terminal

```
f4: [1, 1, 1, 0]
Duljina: 1.73205
v x w = [0, 0, 0, 0]
v . w = 3
union __m128
v * M = [18, 18, 18, 1]
v * M = [14, 15, 16, 0]
v * M = [18, 18, 18, 1]
union __m128
```

3. DXM Matrices

Prijepis 3.1. Functions.h

```cpp
DirectX::XMMATRIX GetInverseMatrix(DirectX::FXMMATRIX mat) {
        DirectX::XMMATRIX mat2 = mat;
        mat2.r[3] = DirectX::XMVECTOR{0,0,0,1};
        DirectX::XMVECTOR Inverse = DirectX::XMMatrixDeterminant(mat2);
        return XMMatrixInverse(&Inverse, mat2);
}

inline DirectX::XMMATRIX* XM_CALLCONV
f( DirectX::FXMMATRIX mat1,
        DirectX::FXMMATRIX mat2,
        DirectX::FXMMATRIX mat3 ){
        std::cout << typeid(mat1).name();
        return new DirectX::XMMATRIX{};
}
```

Prijepis 3.2. Program.cpp

```cpp
        XMFLOAT4X4 f4x4 = { 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 };
        cout << "f4x4: " << ToString(f4x4) << endl;
        XMMATRIX A = XMLoadFloat4x4(&f4x4);
        XMMATRIX B = XMLoadFloat4x4(&f4x4);
        XMMATRIX C = XMMatrixMultiply(A, B);
        XMStoreFloat4x4(&f4x4, C);
        cout << "C = A * B = " << ToString(f4x4) << endl;

        XMMATRIX R = XMMatrixRotationX(PI / 4);
        XMStoreFloat4x4(&f4x4, R);
        cout << "Rotation = " << ToString(f4x4) << endl;

        XMMATRIX I = GetInverseMatrix(R);
        XMStoreFloat4x4(&f4x4, I);
        cout << "Inverse = " << ToString(f4x4) << endl;

        XMMATRIX T = XMMatrixTranspose(R);
        XMStoreFloat4x4(&f4x4, T);
        cout << "Transpose = " << ToString(f4x4) << endl;

        I = XMMatrixTranspose(R);
        XMStoreFloat4x4(&f4x4, I);
        cout << "Identity = " << ToString(f4x4) << endl;
```

- **Is the dynamically created vector 16-byte aligned?**

Nije nužno, jer je tip **struct** XMMATRIX definiran sa **__declspec**(**align**(16)) načinom skladištenja. Alociranje takvog objekta na *heap* memoriji se mora obavljati pomoću **_aligned_malloc**[1] ili nekakve druge metode[2]. Dok je na stogu (na x64 platofrmi) sve poredano po 16 byte blokovima.

DirectXMath.h

```
#ifdef _XM_NO_INTRINSICS_
struct XMMATRIX
#else
__declspec(align(16)) struct XMMATRIX
#endif
{
```

Terminal

```
f4x4:
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15

C = A * B =
56 62 68 74
152 174 196 218
248 286 324 362
344 398 452 506

Rotation =
1 0 0 0
0 0.707107 0.707107 0
0 -0.707107 0.707107 0
0 0 0 1

Inverse =
1 -0 0 0
0 0.707107 -0.707107 0
0 0.707107 0.707107 0
0 0 0 1

Transpose =
1 0 0 0
0 0.707107 -0.707107 0
0 0.707107 0.707107 0
0 0 0 1

Identity =
1 0 0 0
0 0.707107 -0.707107 0
```

[1] https://docs.microsoft.com/en-us/cpp/error-messages/compiler-warnings/compiler-warning-level-3-c4316

[2] https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc

```
0 0.707107 0.707107 0
0 0 0 1
```

4. Precision

Terminal

```
1. length: 0.999999
2. length: 0.99994
3. length: 0.942135
4. length: 0
5. length: 0
6. length: 0
a: 1.0001, b: 1
Compare(a, b) = 1
u and e near equal? true
```

5. DXM Transforms

U ovom zadatku cilj je implementirati cjevovod kao u prošloj vježbi koristeći DirectXMath modul sa njegovim funckijama koje generiraju matrice transofrmacije, te DirectX specifičnim tipovima koji se služe CPU intrizikom za SSE.

Prijepis 5.1. Pipline.h: Primjer implementacije Transform()

```
Model Transform(Model& model, DirectX::CXMMATRIX& transform) {
    Model tfmodel = model;
    for (int i = 0; i < model.GetVertexCount(); i++) {
        DirectX::XMStoreFloat4(
            &(tfmodel.GetVerticesAddress(i)->position),
            DirectX::XMVector3Transform(
                DirectX::XMLoadFloat4(&(tfmodel.GetVerticesAddress(i)->position)),
                transform)
        );
    }
    return tfmodel;
}
```

- **In which space is the object after the last transform?**

U NDC prostoru.

Prijepis 5.2. Primjena inverzne transformacije

```
    XMMATRIX P = XMMatrixPerspectiveFovLH(PI / 3, 40.0f / 25.0f, 0.1, 100);
    DisplayXMMatrix(string("Perspective Transform"), P);
    object = Transform(object, P);
    cout << "Clip objekt: " << object.ToString() << endl;

    object = PerspectiveDivide(object);
    cout << "NDC objekt: " << object.ToString() << endl;

    XMMATRIX D = CreateViewportMatrix(0, 0, 40, 25, 0, 1);
    DisplayXMMatrix(string("Viewport Transform"), D);
    object = Transform(object, D);
    cout << "Screen objekt: " << object.ToString() << endl;

    XMMATRIX M = CreateInverseMatrix(D);
    object = Transform(object, D);
    cout << "NDC objekt: " << object.ToString() << endl;
```

Terminal

```
[ -1 1 0 1 ]
[ 1 1 0 1 ]
[ 1 -1 0 1 ]
[ -1 -1 0 1 ]

World Transform

[ 1.41421 1.41421 0 0 ]
[ -1.41421 1.41421 0 0 ]
[ 0 0 2 0 ]
[ 0 0 0 1 ]
World objekt:
[ -2.82843 -1.19209e-07 0 1 ]
[ -1.19209e-07 2.82843 0 1 ]
[ 2.82843 1.19209e-07 0 1 ]
[ 1.19209e-07 -2.82843 0 1 ]

Camera Transform

[ 1.41421 1.41421 0 0 ]
[ -1.41421 1.41421 0 0 ]
[ 0 0 2 0 ]
[ 0 0 0 1 ]
Camera objekt:
[ -2.82843 -1.19209e-07 6 1 ]
[ -1.19209e-07 2.82843 6 1 ]
[ 2.82843 1.19209e-07 6 1 ]
[ 1.19209e-07 -2.82843 6 1 ]

Perspective Transform

[ 1.08253 0 0 0 ]
[ 0 1.73205 0 0 ]
[ 0 0 1.001 1 ]
[ 0 0 -0.1001 0 ]
Clip objekt:
[ -3.06186 -2.06477e-07 5.90591 6 ]
[ -1.29048e-07 4.89898 5.90591 6 ]
[ 3.06186 2.06477e-07 5.90591 6 ]
[ 1.29048e-07 -4.89898 5.90591 6 ]

NDC objekt:
[ -0.51031 -3.44128e-08 0.984318 1 ]
[ -2.1508e-08 0.816496 0.984318 1 ]
[ 0.51031 3.44128e-08 0.984318 1 ]
[ 2.1508e-08 -0.816496 0.984318 1 ]

Viewport Transform

[ 20 0 0 0 ]
[ 0 -12.5 0 0 ]
[ 0 0 1 0 ]
[ 20 12.5 0 1 ]
Screen objekt:
[ 9.79379 12.5 0.984318 1 ]
[ 20 2.29379 0.984318 1 ]
[ 30.2062 12.5 0.984318 1 ]
[ 20 22.7062 0.984318 1 ]

NDC objekt:
[ 215.876 -143.75 0.984318 1 ]
[ 420 -16.1724 0.984318 1 ]
[ 624.124 -143.75 0.984318 1 ]
[ 420 -271.328 0.984318 1 ]
```