Dynamic Dispatch is the process of deciding which member function to invoke and call at run time. The method has to be declared as virtual. A virtual function table is used by the compiler in C++ to implement dynamic dispatch. To do the same in assembly, you would load the address of the virtual function table, then load the function address, then call the function.

Here is a C++ implementation of Dynamic Dispatch including a Sport class, followed by a Basketball class. Both share the methods getScore() but include their own methods basket and gameOver. First is the Sport class:

```
class Sport{
    public:
        int score=5;


      virtual void getScore(){
        cout<< "The score is: "<< score <<endl;
      }
      virtual void gameOver(){
        if (score == 20){
            cout<< "Game over"<<endl;
        }
      }


};
```

Followed by the Basketball class (go hoos) :

```
class Basketball : public Sport{
public:
    virtual void getScore(){
        cout  << "The Score is : "<<score*2<<endl;

    };B
    virtual int basket(){
        score+=2;
        return score;

    }

};
```

Then the main method, that creates the object and calls the methods

```
int main(){
    Sport *b = new Basketball();
    b -> getScore();
    b -> gameOver();
}
```

Following is some assembly code that replicates the portion of code in which the object is created in the main method:

```
mov     edi, 16
call    operator new(unsigned long)
mov     rbx, rax
mov     QWORD PTR [rbx], 0
mov     DWORD PTR [rbx+8], 0
mov     rdi, rbx
call    Basketball::Basketball() [complete object constructor]
mov     QWORD PTR [rbp-24], rbx
```

And also part of the code that creates the Basketball class that inherits from the Sport class.

```
push    rbp
mov     rbp, rsp
sub     rsp, 16
mov     QWORD PTR [rbp-8], rdi
mov     rax, QWORD PTR [rbp-8]
mov     rdi, rax
call    Sport::Sport() [base object constructor]
mov     edx, OFFSET FLAT:vtable for Basketball+16
mov     rax, QWORD PTR [rbp-8]
mov     QWORD PTR [rax], rdx
nop
leave
ret
```

Thus we can see that generating Dynamic Dispatch in assembly code required calling the initial class in the new class that inherits - as seen in call Sport::Sport() in Basketball. We can also see that for the code to be generated in assembly, first the address of the table is loaded into the object using mov and the base pointer rbp, then the function address into the object, then the call to the function using the call instruction.