Министерство образования Республики Беларусь

Учреждение Образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра электронных вычислительных средств

Лабораторная работа № 4

«ПРОГРАММИРОВАНИЕ НА АССЕМБЛЕРЕ ПРОЦЕССОРА С

АРХИТЕКТУРОЙ INTEL x86»

|  |  |
| --- | --- |
| Проверил: | Выполнили: |
| Порхун М. И. | ст. гр. 850702 |
|  | Маковский Р. А. |
|  | Турко В. Д. |
|  |  |
|  |  |

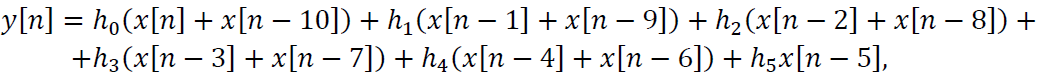
Минск 2020

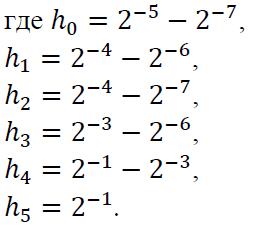
# ЦЕЛЬ РАБОТЫ

Научиться интегрировать ассемблерные вставки в высокоуровневый код; реализовать алгоритм цифровой обработки сигналов средствами ассемблера процессора с архитектурой Intel x86. Научиться оценивать быстродействие программ при помощи внутреннего счетчика тактов процессора.

# ЗАДАНИЕ. ВАРИАНТ 1

Необходимо описать цифровой фильтр двумя способами: на языке C, и с помощью ассемблерной вставки; оценить быстродействие двух разработанных вариантов фильтров. Для демонстрации результата работы Вашего фильтра необходимо построить спектрограммы выходных wav-файлов.





# ХОД РАБОТЫ

\_\_asm

{

rdtsc

mov beg, eax // calculate number of cycles

mov ecx, x // ecx = x (ptr)

add ecx, 4\*10 // ecx += 10

mov ebx, y\_asm //...

add ebx, 4\*10 //...

loop1:

mov eax, [ecx] // eax = x[n]

add eax, [ecx - 4\*10] // eax += x[n - 10]

mov edi, eax

sar eax, 5 // eax >>= 5

sar edi, 7 // edi >>= 7

sub eax, edi

mov [ebx], eax // y[n] = eax

mov eax, [ecx - 4] // eax = x[n - 1]

add eax, [ecx - 4\*9] // eax += x[n - 9]

mov edi, eax

sar eax, 4 // eax >>= 4

sar edi, 6 // edi >>= 6

sub eax, edi

add [ebx], eax // y[n] += eax

mov eax, [ecx - 4\*2] // eax = x[n - 2]

add eax, [ecx - 4\*8] // eax += x[n - 8]

mov edi, eax

sar eax, 4 // eax >>= 5

sar edi, 7 // edi >>= 7

sub eax, edi

add[ebx], eax // y[n] += eax

mov eax, [ecx - 4\*3] // eax = x[n - 3]

add eax, [ecx - 4\*7] // eax += x[n - 7]

mov edi, eax

sar eax, 3 // eax >>= 3

sar edi, 6 // edi >>= 6

sub eax, edi

add[ebx], eax // y[n] += eax

mov eax, [ecx - 4\*4] // eax = x[n - 4]

add eax, [ecx - 4\*6] // eax += x[n - 6]

mov edi, eax

sar eax, 1 // eax >>= 1

sar edi, 3 // edi >>= 3

sub eax, edi

add[ebx], eax // y[n] += eax

mov eax, [ecx - 4\*5] // eax = x[n - 5]

sar eax, 1 // eax >>= 1

add[ebx], eax // y[n] += eax

add ecx, 4 // x++

add ebx, 4 // y\_asm++

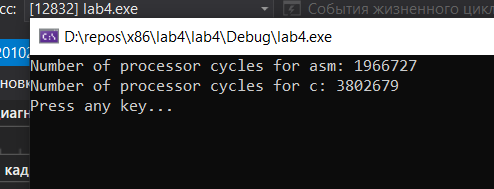
cmp ecx, last // if ecx < last then

jl loop1 // goto cycle;

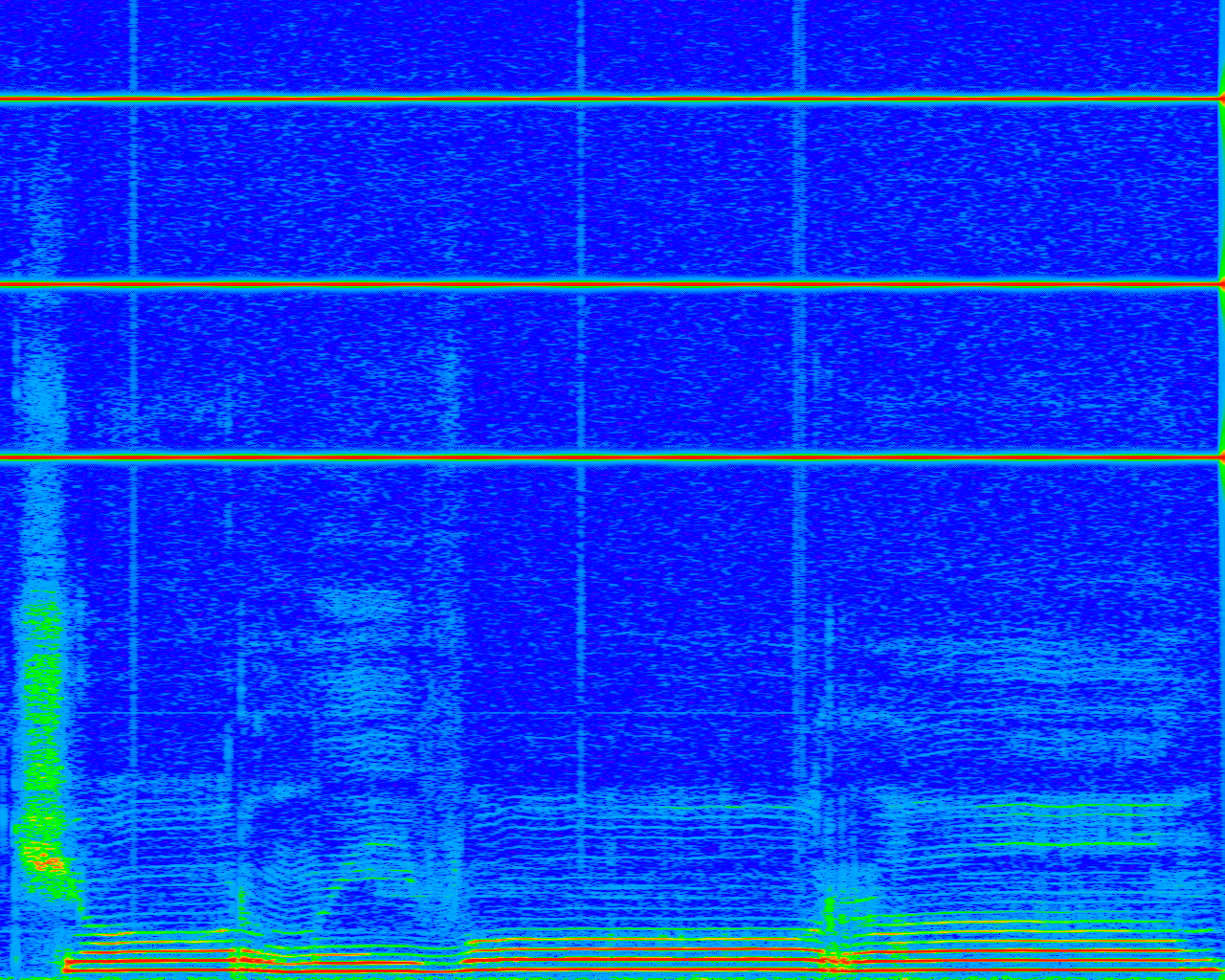
rdtsc // calculate number of cycles

mov end, eax

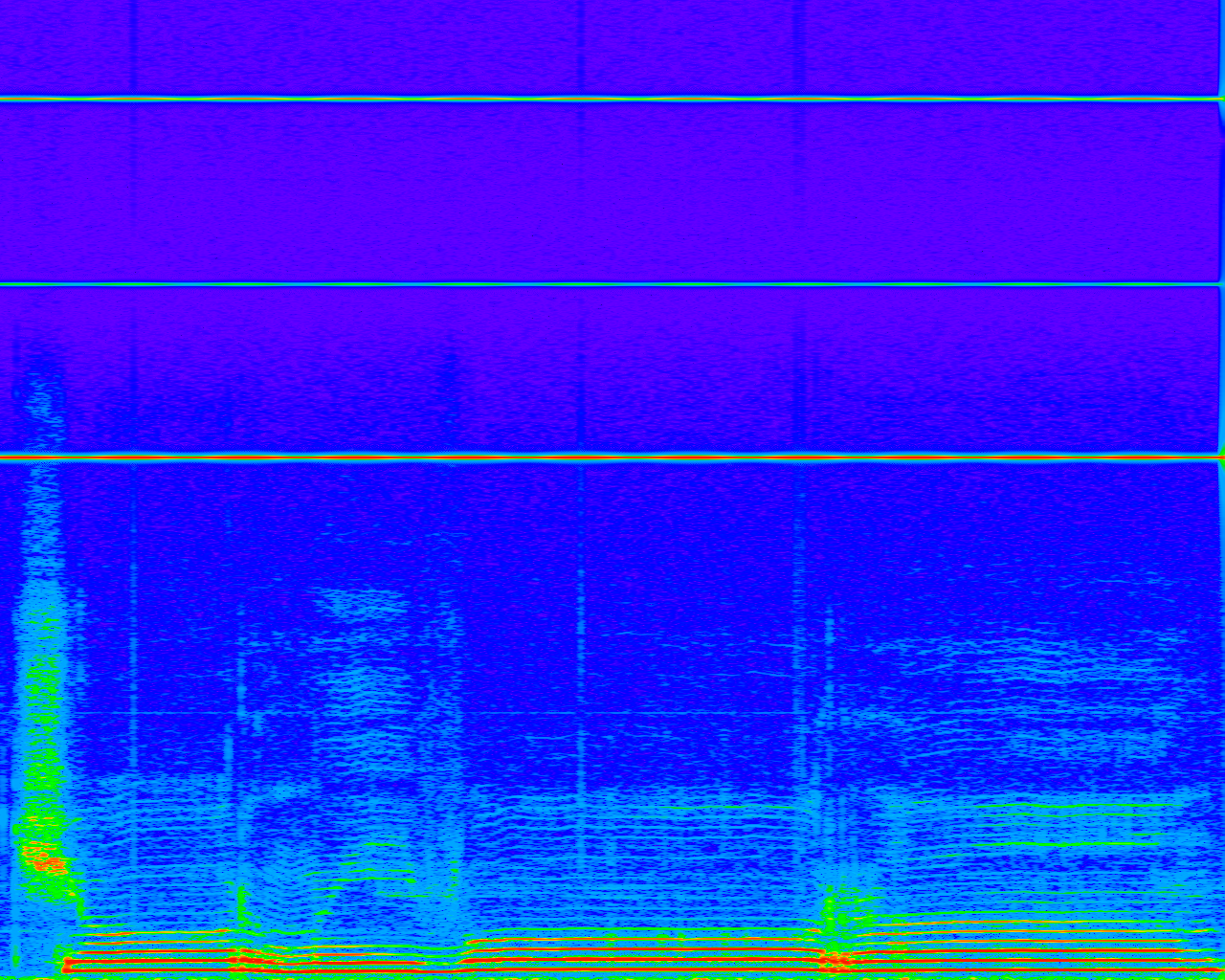
}

****

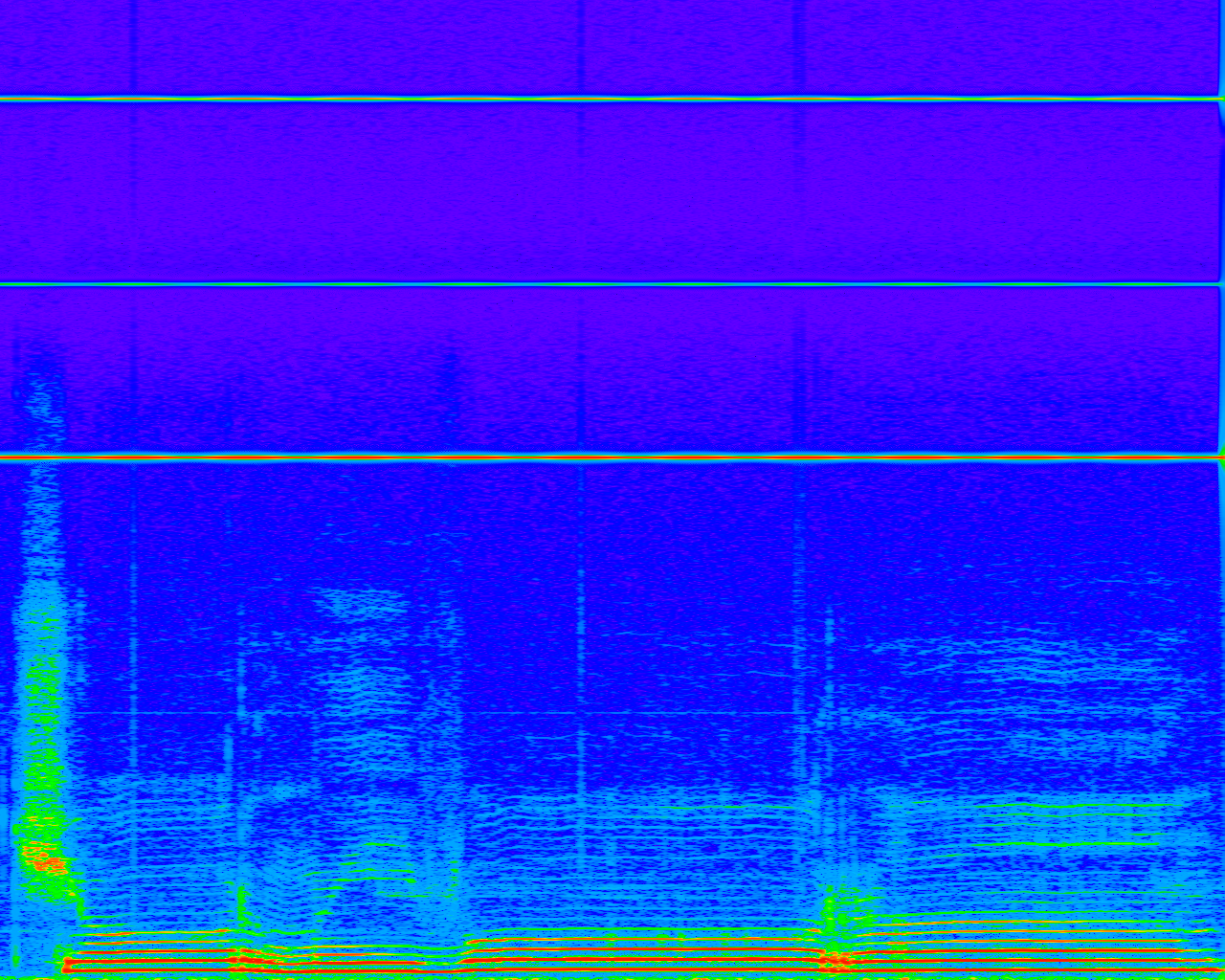
Сравнение работы программы с ассемблерной вставкой и без

****

Спектрограмма исходного файла



Спектрограмма после обработки файла фильтром с низкоуровневой вставкой



Спектрограмма после обработки файла только на высокоуровневом языке C++

# ВЫВОД

В ходе лабораторной работы научились использовать ассемблерные вставки в компиляторе MSVC для Windows на архитектуре Intel x86. Также измерили их эффективность при выполнении на одной машине при помощи счетчика тактов. В результате мы получили разницу почти в 2 раза по количеству тактов затраченных на выполнение одной и той же задачи.