

# Lecture 15 - Performance Tuning Examples

Prof. Brendan Kochunas

October 21, 2020

## Contents

<b>1</b>	<b>Squaring</b>	<b>2</b>
<b>2</b>	<b>Exponentiation</b>	<b>2</b>
<b>3</b>	<b>Repeated Division</b>	<b>3</b>
<b>4</b>	<b>Polynomial Evaluation</b>	<b>4</b>
<b>5</b>	<b>Loop Unrolling</b>	<b>4</b>

These examples are intended to be inspected using the [Compiler Explorer](#). Alternatively they could be done with any Fortran compiler and the `-S` and `-o` options to output the generated assembly.

# 1 Squaring

Use no compiler options. Then use -O3, -Ofast, -Ofast -mavx512f

```
1 real(8) function square(x)
2   implicit none
3   real(8), intent(in) :: x
4   square = x * x * x * x
5   return
6 end function square
7
8
9 real(8) function square2(x)
10  implicit none
11  real(8), intent(in) :: x
12  square2 = x**4
13  return
14 end function square2
15
16 real(8) function square3(x)
17  implicit none
18  real(8), intent(in) :: x
19  square3 = x**(4.0)
20  return
21 end function square3
```

# 2 Exponentiation

Use no compiler options. Then use -O3, -Ofast, -Ofast -mavx512f

```
1 real(8) function exp1(a,b)
2   implicit none
3   real(8), intent(in) :: a,b
4   exp1 = a**b
5   return
6 end function exp1
7
8 real(8) function exp2(a,b)
9   implicit none
10  real(8), intent(in) :: a,b
11  exp2 = EXP(b*LOG(a))
12  return
13 end function exp2
```

### 3 Repeated Division

```
1 subroutine div1(a,b,c,n)
2   implicit none
3   integer, intent(in) :: n
4   real(8), intent(in) :: b(*),c
5   real(8), intent(out) :: a(*)
6
7   integer :: i
8
9   do i=1,n
10    a(i)=b(i)/c
11  end do
12
13  return
14 end subroutine div1
15
16
17 subroutine div2(a,b,c,n)
18   implicit none
19   integer, intent(in) :: n
20   real(8), intent(in) :: b(*),c
21   real(8), intent(out) :: a(*)
22
23   integer :: i
24   real(8) :: rc
25
26   rc=1.0d0/c
27   do i=1,n
28    a(i)=b(i)*c
29  end do
30
31  return
32 end subroutine div2
```

## 4 Polynomial Evaluation

Use no compiler options. Then use -O3, -Ofast, -Ofast -mavx512f

```
1 real(8) function poly1(a0,a1,a2,a3,a4,x)
2   real(8),intent(in) :: a0,a1,a2,a3,a4,x
3   poly1 = a0 + a1*x + a2*x**2 + a3*x**3 + a4*x**4
4   return
5 end function poly1
6
7 real(8) function poly2(a0,a1,a2,a3,a4,x)
8   real(8),intent(in) :: a0,a1,a2,a3,a4,x
9   poly2 = a0 + x*(a1 + x*(a2 + x*(a3 + x*a4)))
10  return
11 end function poly2
```

## 5 Loop Unrolling

Use no compiler options. Then use -O3, -Ofast, -Ofast -mavx512f and -funroll-all-loops

```
1 subroutine lngth1(n,a,s)
2   integer :: n
3   real(8) :: s,a(n)
4   integer :: i
5   s=0.d0
6   do i=1,n
7     s=s+a(i)*a(i)
8   enddo
9 endsubroutine
10
11 subroutine lngth4(n,a,s)
12   integer :: n
13   real(8) :: s,a(n)
14   integer :: i
15   real(8) :: t1,t2,t3,t4
16
17   t1=0.d0; t2=0.d0; t3=0.d0; t4=0.d0
18   do i=1,n-3,4
19     t1=t1+a(i)*a(i)
20     t2=t2+a(i+1)*a(i+1)
21     t3=t3+a(i+2)*a(i+2)
22     t4=t4+a(i+3)*a(i+3)
23   enddo
24   s=t1+t2+t3+t4
25 endsubroutine
```