

Quiz and Homework #1

Making your codes run more efficiently

Dr. John Wallin

George Mason University

January 28, 2010

The Assignment

- ▶ Working alone - answer each of the questions. **Write your answers down!**
- ▶ Discuss the questions with others in class. **Write your revised answer down.**
- ▶ Go home, and do timing tests on at least three of the problems. Use different levels of optimization on the compiler. You should expand the loop sizes and or repeat the calculations with an outer loop to get reliable timing results. (The code should take 10s of seconds to execute.)

The Assignment

Summarize this in on the Wikipage report. For the three problems you solve, you should add.

- ▶ Your name
- ▶ A summary of the problem
- ▶ Your expectations and final conclusions based on discussions.
- ▶ For the problems you solve, your code, the code profile, and the timing results.
- ▶ Any other comments about the code and approach you took to solve the problem, including references.

Question 1

How do I make this code fun faster?

```
do i = 1, 1000
  if (i < 100) then
    a(i) = 10
  else
    a(i) = 20
  endif
enddo
```

Question 2

Which will be faster?

```
if (sqrt(x1**2 + y1**2) < sqrt(x2**2 + y2**2))
```

```
if (x1**2 + y1**2 < x2**2 + y2**2)
```

Question 3

Which code is likely to run faster?

```
do i = 1, 400000
  a(i) = i * exp(i)
enddo
```

could be better written as

```
do i = 1, 400000, 4
  a(i) = i * exp(i)
  a(i+1) = (i+1) * exp(i+1)
  a(i+2) = (i+2) * exp(i+2)
  a(i+3) = (i+3) * exp(i+3)
enddo
```

Question 4

Which code is likely to be more efficient?

```
do i = 1, 3
  a(i) = i * exp(i)
enddo
```

```
a(1) = exp(1)
a(2) = 2*exp(2)
a(3) = 3*exp(3)
```

Question 5

Which code will run faster?

```
do i = 1, 500
  do j = 1,500
    a(i,j) = i * exp(j)
  enddo
enddo
```

```
do j = 1, 500
  do i = 1,500
    a(i,j) = i * exp(j)
  enddo
enddo
```

Question 6

How can this code be written more efficiently?

```
if (x < 10) and (x < 20)
```

Question 7

How can this code be written more efficiently?

```
found = .false.  
do i = 1, large_number  
  if (x(i) == target_value) found = .true.  
enddo
```

Question 8

How can this code be written more efficiently?

```
select case (number)
  case (rarely true)
    do useful stuff here
  case (sometimes true)
    do something else useful here
  case (usually true)
    do the normal thing here
end select
```

Question 9

How can this code be made to run more efficiently?

```
found = .false.
i = 1
do while (i <= large_number .and. .not. found)
  if (value(i) == target_value ) then
    found = .true.
  else
    i = i + 1
  endif
enddo

if (found) then
  ...
```

Question 10

Which of the following will run faster?

```
real :: a(5,100), b(100,5)
do j = 1, 100
  do i = 1, 5
    total = total + a(i,j)
  enddo
enddo
```

```
do j = 1, 5
  do i = 1, 100
    total = total + b(i,j)
  enddo
enddo
```

Question 11

Which of the following will execute faster?

```
increment = xmax /large_number
do i = 1, large_number
  x(i) = i * increment
enddo
```

```
increment = xmax /large_number
sum = increment
do i = 1, large_number
  x(i) = sum
  sum = sum + increment
enddo
```

Question 12

How much faster will the second routine be compared to the first?

```
integer function log2_function( i)
  integer :: i
  log2 = int( log(i) /log(2) )
```

```
integer function log2_function( i)
  integer :: i
  if (i < 2) return 0
  if (i < 4) return 1
  if (i < 8) return 2
  if (i < 16) return 3
  ...
  if (i < 2147483648) return 30
```

Question 14

When should you use the second routine instead of the first routine?

```
value = (1 + x)^3 * cos(x)
```

```
i = (x-xmin) /dx + 1
```

```
value = table(i)
```

Question 15

Why won't this code run efficiently? Can it be fixed?

```
do i = 1, 9999
  a(i) = a(10000 - i) * 5
enddo
```