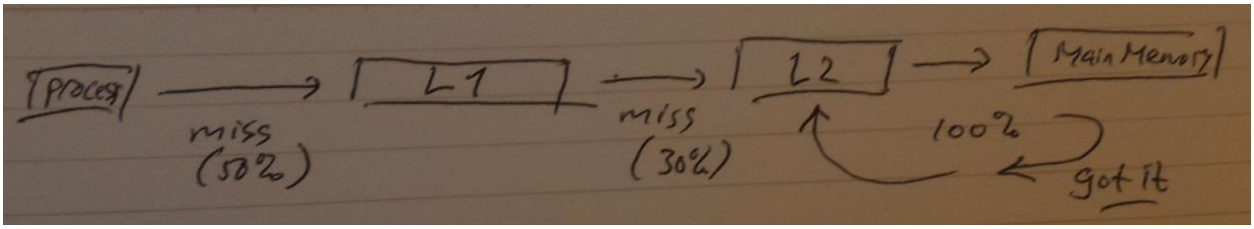
Task 1.1

a. $h_c\ (L2) = (1 - h_c(L1)) * h_c\ (L2) = \frac{50}{100} * \frac{70}{100} =$ 35 %

From Main Memory:



$h_c\ (L2) = (1 - h_c(L1)) * (1 - h_c\ (L2)) * h_c(M) = \frac{50}{100} * \frac{30}{100} =$ 15 %

b. Miss rate = $100\% - h_c(L2) =$ 30%

Miss time (L2) = $(Miss\ rate(L2)) * T_m = 0.3 * 200\ cycles =$ 60 *cycles*

c. Miss rate (L1) = 100% - 50% = 50%

Miss time (L1) = $(1 - h_{L1}) * h_{L2} * T_{L2} + Miss\ time\ (L2) = (0.5 * 0.7 * 15) + 60 =$ 65.25 *cycles*

d. Access Time in Main memory (-10%) = $0.9 * 200 = 180\ cycles$

Miss time (L2) = $(Miss\ rate(L2)) * T_m = 0.3 * 180\ cycles =\ 54\ cycles$

Miss time (L1) = $(1 - h_{L1}) * h_{L2} * T_{L2} + Miss\ time\ (L2) = (0.5 * 0.7 * 15) + 54 =$ 59.25 *cycles*

Improvement = $\frac{65.25 - 59.25}{65.25} * 100\% = 9.3\%$ (6 cycles lesser)

e. L1 -> 75% Hit rate, no L2

Miss time (L1) = $(1 - h_{L1}) * h_m * T_m = 0.25 * 1 * 200 =$ 150 *cycles*

It is possible, but performance-wise and price-wise no. Performance drop reaches almost 3x and it will cost more when it's upgraded rather than adding a second or third level cache

f. $t_{eff} = h_{L1} * T_{L1} + (1 - h_{L1}) * h_{L2} * T_{L2} + (1 - h_{L2}) * h_m * T_m$

$t_{eff} = 0.5 * 2 + (1 - 0.5) * 0.7 * 15 + (1 - 0.7) * 1 * 200$

$t_{eff} = 1 + 0.0525 + 60 =$ 61.0525 *cycles*

Task 1.2

a. Diff #1: CPU caches can be accessed much faster than main memory

Diff #2: CPU caches have much lesser storage capacity than main memory

Cache Line is the way the data in the cache is organized. When a process requests a certain data from cache, the entire data in that cache line, where the process access the data, is brought up to the process.

b.

```
for i = 0 to n do
    read B(i,j)                 //[1]access cache
    for j = 0 to n do
        read A(j,i)                 //[2]access memory
        B (i,j) += A (j,i)          //arithmetic operation
    end for
    write B(i,j)                //[3]write to cache
```

Access is not consecutive, Access[2] jumps from 1 cache line to the other.

c. $T(n) = 2nT_c + n^2(T_a + T_m)$

d. If Matrix A is stored column wise, matrix A will only fetch 1 cache line for all iteration in inner loop.
$T(n) = 2nT_c + n(nT_a + T_m)$

e. Row-wise Matrix A $= 2nT_c + n^2(T_a + T_m) = 16nT_a + n^2(T_a + 80T_a)$
$= nT_a\ (16 + 81n)$
Column-wise Matrix A $= 2nT_c + n(nT_a + T_m) = 16nT_a + n(nT_a + 80T_a)$
$= nT_a\ (96 + n)$

Comparison $= \frac{T(n) for\ row-wise\ M_A}{T(n) for\ col-wise\ M_A} = \frac{81n+16}{n+96} \approx 80$ x

Which means, if M_A is stored column-wise, it will be 80x faster

Task 1.3

a. $Speedup = \frac{T_{total}(1)}{\frac{T_{parallel}(1)}{p} + T_{serial}(1)}$

$Speedup = \frac{T_{total}(1)}{\frac{(1-\alpha)T_{total}(1)}{p} + \alpha T_{total}(1)}$ with α = the part of code that can't be parallelized

$Speedup = \frac{T_{total}(1)}{\frac{(1-0.1)T_{total}(1)}{16} + 0.1T_{total}(1)}$

$Speedup = \frac{T_{total}(1)}{0.05625.T_{total}(1) + 0.1T_{total}(1)} = \frac{T_{total}(1)}{0.15625.T_{total}(1)} =$ 6.4 x

b. $10 = \frac{T_{total}(1)}{\frac{(1-\alpha)T_{total}(1)}{16} + \alpha T_{total}(1)} = \frac{T_{total}(1)}{\frac{(1-\alpha)T_{total}(1) + 16.\alpha.T_{total}(1)}{16}} = \frac{16.T_{total}(1)}{(1-\alpha)T_{total}(1) + 16.\alpha.T_{total}(1)}$

$10 = \frac{16.T_{total}(1)}{(1-\alpha+16\alpha)T_{total}(1)} = \frac{16}{(1-\alpha+16\alpha)}$

$10(1+15\alpha) = 16$

$1 + 15\alpha = 1.6$

$15\alpha = 0.6\ ;\ \alpha = 0.04$

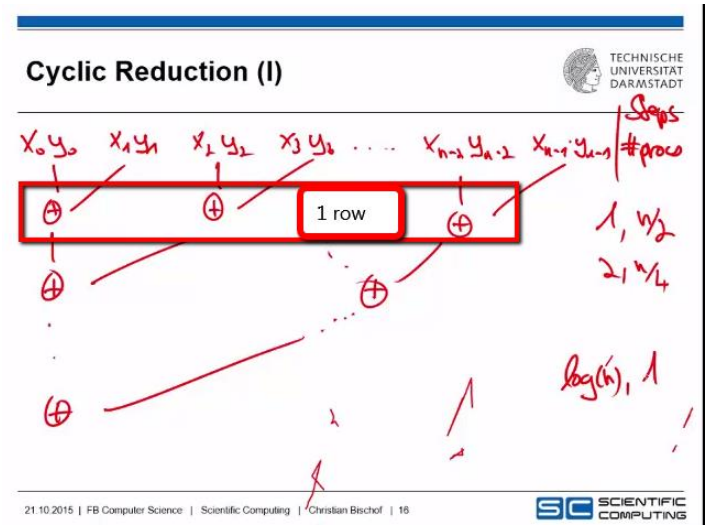
Which means, 96% of the code must be able to operate parallel (only 4% is allowed for serial process)

Task 1.4

a. Speedup means how much faster can the code run in parallel compared in serial form
Efficiency means the comparison between the speedup obtained in parallelism with the number of processors used

b.
```
r = MIN_INT                          //r=result
for i = 0 to n do
    read (x[i])
    if (max(r,x[i]) == x[i])         //max function -> k
        r = x[i]                     //arithmetic opr -> k
    endif
end for
write(r)
```

Steps required =$2kn$, where k = the arithmetic operation and max function

c.


Each processor will require $2k$steps
By taking that $\sqrt[2]{p} \in R$ and $p = \frac{n}{2}$, The number of processors required every row*-$n = \log_2 n$
Total steps = $2k \log_2 n$
**1row is 1 calculation cycle*

d. $Speedup = \frac{2kn}{2k \log_2 n} = \frac{n}{\log_2 n} = \log_n 2^n = \log_{2p} 2^{2p} = \log_{2p} 4^p = p.\log_{2p} 4$

$Efficiency = \frac{Speedup}{p} = \log_{2p} 4$

e. With $n = p^2$;

$Speedup = \frac{p^2}{\log_2 p^2} = \frac{1}{2} p^2.\log_p 2$

$Efficiency = \frac{1}{2} p.\log_p 2$