Texas A&M University

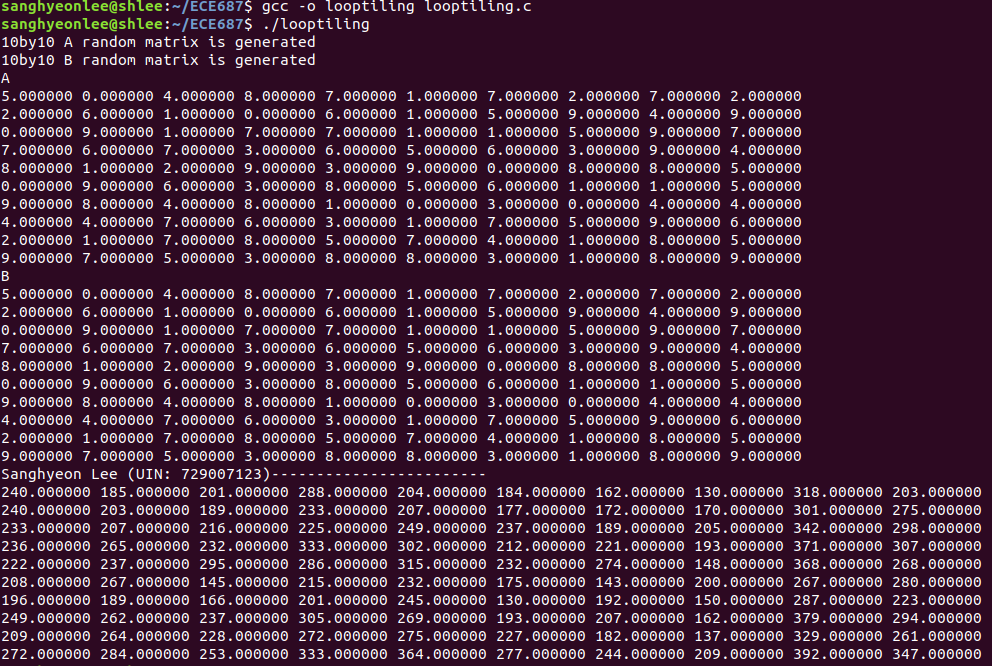
Homework 5 ECEN 687

Introduction to VLSI Design Automation

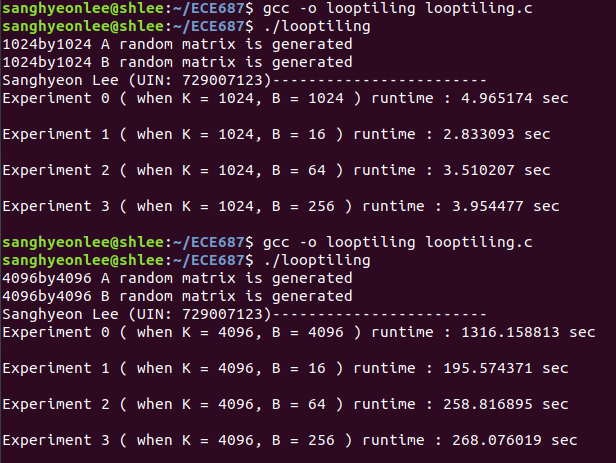
Sanghyeon Lee (UIN: 729007123)

Index

1. Program test
2. Experiment results and review
3. Code
4. Program test when k = 10



1. Parameters k = 1024, B[1,2,3] = 16, 64, 256, k = 4096, B[1,2,3] = 16, 64, 256



B (block size) is batch size which means the loop in the program only calculate with batch size and add them. When matrix A and B has 10x10 size and batch size is 10, they need to calculate 10^3. However, when batch size is 5, they only need to calculate 5^3 \* (10/5)^2. So, by changing the parameter B, we can save resources. Thus, when B is increased, the CPU runtime will also be increased by my formula B^3 \* (k/B)^2) <= k^3 (for B<=k)

1. Code

|  |
| --- |
| 1. #include <stdio.h> 2. #include <stdlib.h> 3. #include <time.h> 4. #define k 1024 5. float (\*MakeRandom\_kXk\_Array())[k]; 6. //k by k random matrix A generation 7. void printfunc(float (\*arr)[k]); 8. //matrix printing funcion 9. float (\*Experiment\_1(float (\*A)[k], float (\*B)[k]))[k]; 10. //Matmul function without using loop tiling 11. float (\*Experiment\_2(float (\*A)[k], float (\*B)[k], int B0))[k]; 12. //Matmul function using lopp tiling 13. int main() 14. { 15. int B1 = 16; 16. int B2 = 64; 17. int B3 = 256; 18. float (\*a)[k] = MakeRandom\_kXk\_Array(); 19. float (\*b)[k] = MakeRandom\_kXk\_Array(); 20. clock\_t start0, start1, start2, start3, end0, end1, end2, end3; 21. float expt0, expt1, expt2, expt3; 22. //CPU time check variables 24. printf("%dby%d A random matrix is generated\n",k,k); 25. printf("%dby%d B random matrix is generated\n",k,k); 26. //printf("A\n"); 27. //printfunc(a); 28. //printf("B\n"); 29. //printfunc(b); 30. printf("Sanghyeon Lee (UIN: 729007123)------------------------\n"); 32. start0 = clock(); 33. float (\*y0)[k] = Experiment\_1(a, b); 34. end0 = clock(); 35. expt0 = (float)(end0 - start0)/CLOCKS\_PER\_SEC; 36. //printf("Experiment 1) With tiling\nA\*B=Y:\n"); 37. //printfunc(y0); 39. printf("Experiment 0 ( when K = %d, B = %d ) runtime : %f sec\n\n", k, k, expt0); 41. start1 = clock(); 42. float (\*y1)[k] = Experiment\_2(a, b, B1); 43. end1 = clock(); 44. expt1 = (float)(end1 - start1)/CLOCKS\_PER\_SEC; 45. printf("Experiment 1 ( when K = %d, B = %d ) runtime : %f sec\n\n", k, B1, expt1); 47. start2 = clock(); 48. float (\*y2)[k] = Experiment\_2(a, b, B2); 49. end2 = clock(); 50. expt2 = (float)(end2 - start2)/CLOCKS\_PER\_SEC; 51. printf("Experiment 2 ( when K = %d, B = %d ) runtime : %f sec\n\n", k, B2, expt2); 53. start3 = clock(); 54. float (\*y3)[k] = Experiment\_2(a, b, B3); 55. end3 = clock(); 56. expt3 = (float)(end3 - start3)/CLOCKS\_PER\_SEC; 57. printf("Experiment 3 ( when K = %d, B = %d ) runtime : %f sec\n\n", k, B3, expt3); 59. return 0; 60. } 61. float (\*MakeRandom\_kXk\_Array())[k] 62. { 63. static float arr[k][k]; 64. for(int i = 0; i<k;i++) 65. { 66. for (int j = 0; j<k;j++) 67. { 68. arr[i][j] = rand()%k; 69. } 70. } 71. return arr; 72. } 73. void printfunc(float (\*arr)[k]) 74. { 75. for(int i = 0; i<k;i++) 76. { 77. for (int j = 0; j<k;j++) 78. { 79. printf("%3f ",arr[i][j]); 80. } 81. printf("\n"); 82. } 83. } 84. float (\*Experiment\_1(float (\*A)[k], float (\*B)[k]))[k] 85. { 86. float sum1 = 0; 87. static float y[k][k]; 89. for(int i=0;i<k;i++) 90. { 91. for(int j=0;j<k;j++) 92. { 93. sum1 = 0; 94. for(int m=0;m<k;m++) 95. { 96. sum1 += A[i][m]\*B[m][j]; 97. } 98. y[i][j] = sum1; 99. } 100. } 101. return y; 102. } 103. float (\*Experiment\_2(float (\*A)[k], float (\*B)[k], int B0))[k] 104. { 105. float sum2 = 0; 106. int B1 = B0; 107. int B2 = B0; 108. int B3 = B0; 110. static float y[k][k]; 112. for(int ii=0;ii<k;ii+=B1) 113. { 114. for(int jj=0;jj<k;jj+=B2) 115. { 116. for(int mm=0;mm<k;mm+=B3) 117. { 118. for(int i=ii; i<ii+B1;i++) 119. { 120. for(int j=jj;j<jj+B2;j++) 121. { 122. sum2 = 0; 123. for(int m=mm;m<mm+B3;m++) 124. { 125. sum2 += A[i][m]\*B[m][j]; 126. } 127. y[i][j] += sum2; 128. } 129. } 130. } 131. } 132. } 133. return y; 134. } |