學號:B06902136
系級:資工四
姓名:賴冠毓

Part 1: Handwritten

2.8

f: x5, g: x6, h: x7
i: x28, j: x29
A[]: x10
B[]: x11

```
addi x30, x10, 8    //x30 = &(A[1])
addi x31, x10, 0    //x31 = &A
sd x31, 0(x30)      //A[1] = x31 = &A
ld x30, 0(x30)      //x30 = A[1] = &A
add x5, x30, x31    //f = &A + &A = 2(&A)
```

⇨ C code: f = 2(&A);

2.9

addi x30, x10, 8:

type	immediate	rs1	funct3	rd	opcode
I-type	000000001000	01010	000	11110	0010011

addi x31, x10, 0:

type	immediate	rs1	funct3	rd	opcode
I-type	000000000000	01010	000	11111	0010011

sd x31, 0(x30):

type	imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
S-type	0000000	11111	11110	011	00000	0100011

ld x30, 0(x30):

type	immediate	rs1	funct3	rd	opcode
I-type	000000000000	11110	011	11110	0000011

add x5, x30, x31:

type	funct7	rs2	rs1	funct3	rd	opcode
R-type	0000000	11111	11110	000	00101	0110011

2.16

register 總數變 4 倍 => register 的 index 在 2 進位表示下位數增加 2 位

instruction 總數變 4 倍 => operation 的 index 在 2 進位表示下位數增加 2 位

2.16.1

R-type:

rd, rs1, rs2: 5 bits 變 7 bits

opcode: 7 bits 變 9 bits

2.16.2

I-type:

rd, rs1: 5 bits 變 7 bits

immediate: bit 數不變，所以可能的 offset 不變

opcode: 7 bits 變 9 bits

2.16.3

decrease: register 總數變多，避免 register 不夠用而需要多進行操作，讓指令的總數變少，program size 可能變小

increase: 單一指令長度增加，program size 可能變大

Part 2: Report on matrix multiplication

the naive matrix multiplication:

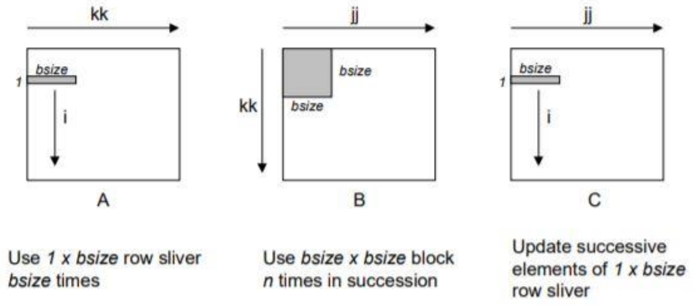
1.cycles: 16892677

2.load and store:

$\text{load}: 128 * 128 * 128 * 3 = 6291456$

$\text{store}: 128 * 128 * 128 = 2097152$

3.keep registers being used as much as possible before they’ re replaced: blocking

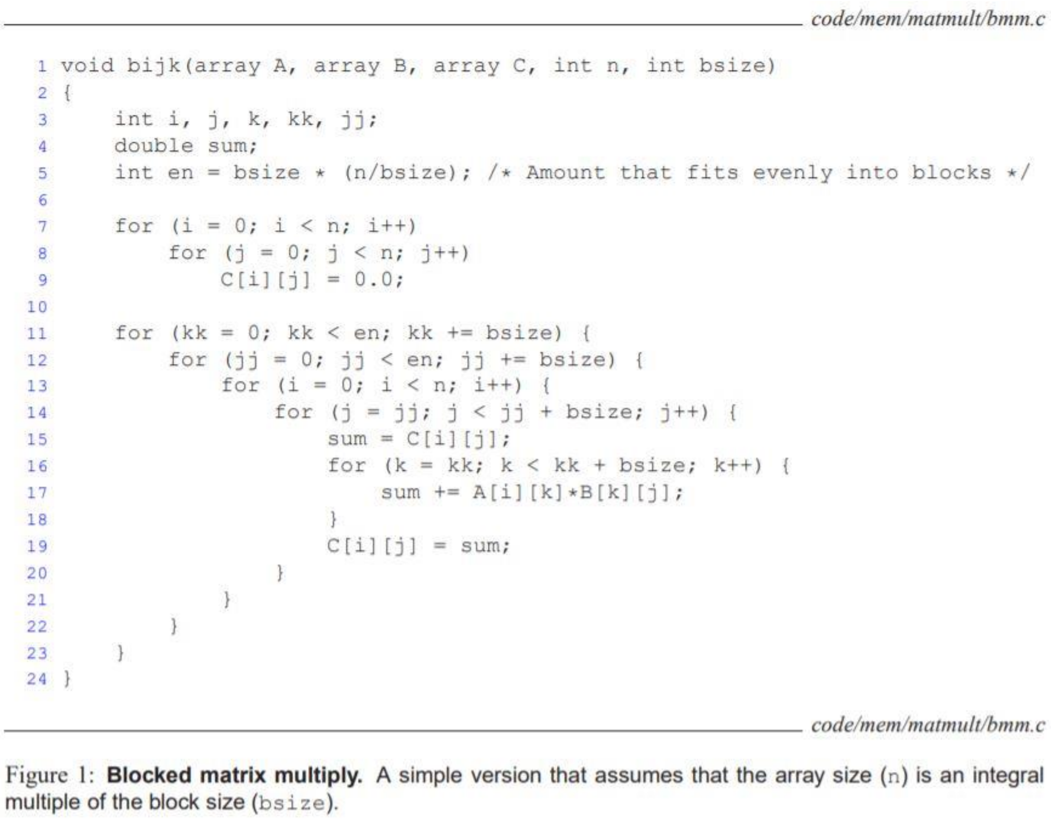


Use *1 x bsize* row sliver *bsize* times

Use *bsize x bsize* block *n* times in succession

Update successive elements of *1 x bsize* row sliver

Figure 2: **Graphical interpretation of blocked matrix multiply** The innermost (j, k) loop pair multiplies a $1 \times$ *bsize* sliver of A by a *bsize* $\times$ *bsize* block of B and accumulates into a $1 \times$ *bsize* sliver of C.



code/mem/matmult/bmm.c

```c
void bijk(array A, array B, array C, int n, int bsize)
{
    int i, j, k, kk, jj;
    double sum;
    int en = bsize * (n/bsize); /* Amount that fits evenly into blocks */

    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            C[i][j] = 0.0;

    for (kk = 0; kk < en; kk += bsize) {
        for (jj = 0; jj < en; jj += bsize) {
            for (i = 0; i < n; i++) {
                for (j = jj; j < jj + bsize; j++) {
                    sum = C[i][j];
                    for (k = kk; k < kk + bsize; k++) {
                        sum += A[i][k]*B[k][j];
                    }
                    C[i][j] = sum;
                }
            }
        }
    }
}
```

code/mem/matmult/bmm.c

Figure 1: **Blocked matrix multiply.** A simple version that assumes that the array size (n) is an integral multiple of the block size (bsize).

4. loop controls:

$128 * 128 * 128 * 4 = 83388608$

Reference: https://csapp.cs.cmu.edu/public/waside/waside-blocking.pdf?fbclid=IwAR0cQzCbIAqbLHp8UyykaKEd7YVn0p25BeALc50ZsbJbdBy1hjC5yn1TKTc