Compsys - A6

Christian R (srw812), Nicolai N (dwx635) og Andreas J (cmn945)

January 9, 2022

Pipeline

1.1

1.2

1.3

1.4

1.5

Assembler

2.1

Registers

- rsp: pointer
- r11: long
- rax: long
- rdx: long
- rdi: pointer

2.2

There is no call instructions, however there is a single return instruction, namely "ret %r11", which returns the %r11 register.

2.3

The function p takes one argument (%rdi) and this value is assigned to a long. A condition is then encountered, if this is true, it will perform a conditional jump. If false, it will loop until true by an unconditional jump. Finally, it will return a long

2.4

The conditional statement in line 8 "cbe $0, %rax, .L4" means that if %rax is equal to 0 jump to L4 otherwise, it enters a loop that jumps back to the condition until it is true. This can be translated to a while-loop.

2.5

```
long p(long *arg){
    retval = 0;
    a = *arg;
    while (a <= 0){
        retval = retval + a;
        arg++;
        a = *arg;
    }
    return retval;
}
```

Cache

3.1

Since we constantly move between the two arrays as we multiply them the stride is anywhere from the size of A (as in the case of the top left element of C where we multiply the first element of A with the first element of B) to the size of A + the size of B (as in the case of thee top right element of C where we multiply the first element of A with the last element of B)

3.2

For the reason above We would say that the program has poor spatial locality since with each instruction we are quite far from the next value to be loaded and thus have to jump back and forth a lot eventually potentially leading to overriding a cache slot which we need again soon if the size of n and m are large
In terms of temporal locality it's not too bad in terms of A as every other instruction we access the same value from A m times at which point we are done with that value and then go to the next one. With B however it is quite poor again. We access each element in B n times but we go through the entire array only to jump all the way back to the start again and start over at which point the first value of B very well could have left the cache already.

3.3

A possible way to improve locality for the program would be to handle each array separately by first going through A loading the first element into all spaces in the first row of C, the second element into second row

etc. From there we then go through B and multiply the values individually into each column of C.
This greatly improves spatial locality since we go through each array separately for each element meaning we achieve of stride of sizeof(int) which is the smallest possible stride we can achieve with the data type off our arrays.
There is also a significant improvement to temporal locality since, once we start using a value from memory, we make sure we are completely finished with it before we move onto the next value