Mohit Khatwani, AJ75499

[CMSC-611] Advanced Computer Architecture: HW3

1) Given: Cache Size = 256KB, Block Size = 32 Bytes, Address size = 32 bits, calculate the size of the tag, offset and index fields and explain very briefly how you computed your values, for

a. Direct Mapped cache

Index in direct mapped cache depends on number of blocks in the cache.

Number of blocks = $\frac{Cache\ Size}{Block\ size}$ = $\frac{256*1024}{32}$ = 8192 blocks. = 2^{13}

So, index is 13 bits.

We have 32 bytes block size. So, offset is 32 = 2^5. Offset can be represented in 5 bits.

Address size = tags + index + offset

tags = 32 – 13 – 5 = 14 bits.

b. 8 – way Associative Cache.

In set-associative cache index bits are represented by number of sets.

Number of sets = $\frac{No.of\ lines}{k-sets}$ = $\frac{8192}{8}$ = 1024 = 2^{10}

Offset is 5 bits, as calculated before.

Address size = tags + index + offset

Tags = 32 – 10 – 5 = 17 bits.

c. Fully Associative Cache

As there is only 1 set, no bits are required for index.

Offset is same as before, 5 bits.

Tags = 32 – 5 = 27 bits.

2) Consider the following code, which multiplies two vectors that contain single-precision complex values:

```
for (i=0;i<300;i++) {
        c_re[i] = a_re[i] * b_re[i] – a_im[i] * b_im[i];
        c_im[i] = a_re[i] * b_im[i] + a_im[i] * b_re[i];
}
```

Assume that the processor runs at 700 MHz and has a maximum vector length of 64. The load/store unit has a start-up overhead of 15 cycles; the multiply unit, 8 cycles; and the add/subtract unit, 5 cycles.

a) What is the arithmetic intensity of this kernel? Justify your answer.

This code reads four floats and writes two floats for every six FLOPS, so arithmetic intensity is 6/6 = 1

b) Convert this loop into VMIPS assembly code using strip mining.

```
        Here maximum vector length given is 64.
        LI $VL, 44                  #perform the first 44 ops
        LI $R1, 0                   #initialize the index
loop:   LV $V1, a_re+$R1            #Load a_re
        LV $V3, b_re+$R1            #Load b_re
        MULVV.S $v5,$v1,$v3         #a_re * b_re
        LV $V2,a_im+$R1             #Load a_im
        LV $V4,b_im+$r1             #Load b_im
        MULVV.S    $V6,$V2,$V4      #a_im * b_im
        SUBVV.S    $V5,$V5,$V6      # a_re * b_re - a_im * b_im
        SV         $V5,c_re+$r1     #store c_re
        MULVV.S    $V5,$V1,$V4      #a_re * b_im
        MULVV.S    $V6,$V2,$V3      #a_im * b_re
        ADDVV.S    $V5,$V5,$V6      # a_re * b_im + a_im * b_re
        SV         $V5,c_im+$r1     #store in c_im
        BNE        $R1,0,else       #check if its first iteration
        ADDI       $r1,$r1,#44      #first iteration, increment by 44
        JUMP loop
else:   ADDI $R1, $R1, #256         #not first iteration
```

c) Assuming chaining and a single memory pipeline, how many chimes are required? Write done the detailed code to justify your answer.

```
1. MULVV.S    LV       # a_re*b_re, load a_im
2. LV         MULVV.S  #load b_im, a_im*b_im
3. SUBVV.S    SV       #sub and store c_re
4. MULVV.S    LV       #a_re*b_im, load a_re
5. MULVV.S    LV       #a_im *b_re, load b_re
6. ADDVV.S    SV       #add and store c_im
```

6 chimes are required

d) If the vector sequence is chained, how many clock cycles are required per complex result value, including overhead?

Total cycles per iteration = 6 chimes * 64 elements + 15 cycles (load/store) * 6 + 8 cycles (multiply) * 4 + 5 cycles (add/sub) * 2 = 516

Cycles per result = $\frac{Total\ cyles\ per\ instruction}{Number\ of\ results} = \frac{516}{128} = 4$