Week 6

Flexible MatMul

- Transpose: (K^T)

```
if (transposed == 0) vse32_v_f32m1(&o[i * dim3 + j], partial_sum, vl);
else vsse32_v_f32m1(&o[j * dim1 + i], dim1 * 4, partial_sum, vl);
```

- Bias & Scale

```
for (int k = 0; k < dim2; k++) {
  vfloat32m1_t b_vec = vle32_v_f32m1(&mat_b[k * dim3 + j], vl);
  partial_sum = vfmacc_vf_f32m1(partial_sum, mat_a[i * dim2 + k], b_vec, vl);
}

vfloat32m1_t bias_vec = vle32_v_f32m1(&bias[i * dim3 + j], vl);
partial_sum = vfadd_vv_f32m1(partial_sum, bias_vec, vl);
```

- Possible improvements:
  - Use higher length multiplier (m2, m4)

Softmax

- Online normalization
  - Only need to load data twice

$$e^{a-b} \times e^{b-c} = e^{a-c}$$

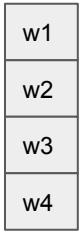
1. $m_0 \leftarrow -\infty$
2. $d \leftarrow 0$
3. **for** j $\leftarrow 1, V$ **do**
4. $\quad m_j \leftarrow \text{IntMax}(m_{j-1}, x_j)$
5. $\quad d \leftarrow d \gg (m_j - m_{j-1}) + 2^{x_j - m_j}$
6. $\quad y_i \leftarrow 2^{x_j - m_j}$
7. **end for**
8. **for** i $\leftarrow 1, V$ **do**
9. $\quad y_i \leftarrow \frac{y_i \gg (m_V - m_i)}{d}$
10. **end for**

```
// ================================================
// update max and sum
// ================================================

// initialize cur_sum and cur_max
cur_sum = vfmv_v_f_f32m1(0, vl);
cur_max = vfmv_v_f_f32m1(-99, vl);

for (int j = 0; j < col; j++) {
  // x
  vfloat32m1_t vec = vlse32_v_f32m1(&mat[i * col + j], col * 4, vl);

  // m[j] = max(m[j-1], x)
  next_max = vfmax_vv_f32m1(cur_max, vec, vl);
  // vec = exp(x - m[j])
  vec = vfsub_vv_f32m1(vec, next_max, vl);
  vec = __exp_2xf32(vec, vl);

  // tmp_vec = exp(m[j-1] - m[j])
  tmp_vec = vfsub_vv_f32m1(cur_max, next_max, vl);
  tmp_vec = __exp_2xf32(tmp_vec, vl);

  // sum[j] = exp(x - m[j]) + sum[j-1] * exp(m[j] - m[j-1])
  next_sum = vfmul_vv_f32m1(cur_sum, tmp_vec, vl);
  next_sum = vfadd_vv_f32m1(next_sum, vec, vl);

  cur_sum = next_sum;
  cur_max = next_max;
}
```

Softermax: Hardware/Software Co-Design of an Efficient Softmax for Transformers,Jacob R. Stevens (Purdue) Rangharajan Venkatesan Steve Dai Brucek Khailany Anand Raghunathan (Purdue)

Attention Heads Concatenation

- Q[i] = x * Wq[i], Wq[i] (d_model x dk), dk = d_model / head
  - In PyTorch, Wq[1] ~ Wq[h] horizontally concatenated.
  - In C, vertically concatenated (continuous memory address)




```
// ================================================
// Concatenate all the heads
// ================================================

float mhsa_2[n * d_model] __attribute__((aligned(32 * NR_LANES)));

for (int k = 0; k < h; k++) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < dk;) {
      int vl = vsetvlmax_e32m1();
      if (j + vl > dk) vl = vsetvl_e32m1(dk - j);

      vfloat32m1_t vec = vle32_v_f32m1(&mhsa_1[k * n * dk + i * dk + j], vl);
      vse32_v_f32m1(&mhsa_2[d_model * i + k * dk + j], vec, vl);

      j += vl;
    }
  }
}
```

Error Detection

- UOP (0.0001, 5 bits)

  3.1415: 01000000010010010000111001010110

  3.1514: 01000000010010010000110010110011

- Relative Difference

```
float diff;
if (mat_b[idx] == 0) diff = mat_a[idx];
else diff = (mat_a[idx] - mat_b[idx]) / mat_b[idx];

if (diff > 0.01 || diff < -0.01) {
```

Some Problems

1. Spike doesn’t support large data (64 x 64 OK, 64 x 256 NOT)
2. How to choose LMUL (m1, m2, m4, m8)
3. Hardware simulation automatically exits after running several instructions