#ifndef GEMM\_H

#define GEMM\_H

#include "activations.h"

#include <stdint.h>

#include <stddef.h>

#ifdef \_\_cplusplus

extern "C" {

#endif

void convolution\_2d(int w, int h, int ksize, int n, int c, int pad, int stride,

float \*weights, float \*input, float \*output, float \*mean);

static inline void set\_bit(unsigned char \*const dst, size\_t index) {

size\_t dst\_i = index / 8;

int dst\_shift = index % 8;

dst[dst\_i] |= 1 << dst\_shift;

//dst[dst\_i] |= 1 << (8 - dst\_shift);

}

static inline unsigned char get\_bit(unsigned char const\*const src, size\_t index) {

size\_t src\_i = index / 8;

int src\_shift = index % 8;

unsigned char val = (src[src\_i] & (1 << src\_shift)) > 0;

//unsigned char val = (src[src\_i] & (1 << (8 - src\_shift))) > 0;

return val;

}

int is\_avx();

int is\_fma\_avx2();

void float\_to\_bit(float \*src, unsigned char \*dst, size\_t size);

void transpose\_block\_SSE4x4(float \*A, float \*B, const int n, const int m,

const int lda, const int ldb, const int block\_size);

void transpose\_bin(uint32\_t \*A, uint32\_t \*B, const int n, const int m,

const int lda, const int ldb, const int block\_size);

void gemm\_nn\_custom\_bin\_mean\_transposed(int M, int N, int K, float ALPHA\_UNUSED,

unsigned char \*A, int lda,

unsigned char \*B, int ldb,

float \*C, int ldc, float \*mean\_arr);

void im2col\_cpu\_custom(float\* data\_im,

int channels, int height, int width,

int ksize, int stride, int pad, float\* data\_col);

void im2col\_cpu\_custom\_align(float\* data\_im,

int channels, int height, int width,

int ksize, int stride, int pad, float\* data\_col, int bit\_align);

void im2col\_cpu\_custom\_bin(float\* data\_im,

int channels, int height, int width,

int ksize, int stride, int pad, float\* data\_col, int bit\_align);

void im2col\_cpu\_custom\_transpose(float\* data\_im,

int channels, int height, int width,

int ksize, int stride, int pad, float\* data\_col, int ldb\_align);

void activate\_array\_cpu\_custom(float \*x, const int n, const ACTIVATION a);

void transpose\_32x32\_bits\_reversed\_diagonale(uint32\_t \*A, uint32\_t \*B, int m, int n);

void gemm\_bin(int M, int N, int K, float ALPHA,

char \*A, int lda,

float \*B, int ldb,

float \*C, int ldc);

void repack\_input(float \*input, float \*re\_packed\_input, int w, int h, int c);

void convolution\_repacked(uint32\_t \*packed\_input, uint32\_t \*packed\_weights, float \*output,

int w, int h, int c, int n, int size, int pad, int new\_lda, float \*mean\_arr);

void gemm\_nn\_bin\_32bit\_packed(int M, int N, int K, float ALPHA,

uint32\_t \*A, int lda,

uint32\_t \*B, int ldb,

float \*C, int ldc, float \*mean\_arr);

void transpose\_uint32(uint32\_t \*src, uint32\_t \*dst, int src\_h, int src\_w, int src\_align, int dst\_align);

void gemm\_nn\_bin\_transposed\_32bit\_packed(int M, int N, int K, float ALPHA,

uint32\_t \*A, int lda,

uint32\_t \*B, int ldb,

float \*C, int ldc, float \*mean\_arr);

void forward\_maxpool\_layer\_avx(float \*src, float \*dst, int \*indexes, int size, int w, int h, int out\_w, int out\_h, int c,

int pad, int stride, int batch);

void gemm(int TA, int TB, int M, int N, int K, float ALPHA,

float \*A, int lda,

float \*B, int ldb,

float BETA,

float \*C, int ldc);

void gemm\_cpu(int TA, int TB, int M, int N, int K, float ALPHA,

float \*A, int lda,

float \*B, int ldb,

float BETA,

float \*C, int ldc);

#ifdef GPU

void gemm\_ongpu(int TA, int TB, int M, int N, int K, float ALPHA,

float \*A\_gpu, int lda,

float \*B\_gpu, int ldb,

float BETA,

float \*C\_gpu, int ldc);

void gemm\_gpu(int TA, int TB, int M, int N, int K, float ALPHA,

float \*A, int lda,

float \*B, int ldb,

float BETA,

float \*C, int ldc);

#endif

#ifdef \_\_cplusplus

}

#endif

#endif