La BitMap è stata implementata per mappare blocchi liberi e occupati.

Per l’implementazione della BitMap sono state utilizzate due struct.

La prima struct si occupa della BitMap vera e propria e i campi che la caratterizzano. Sono importanti infatti il numero dei bit , il numero dei blocchi e le entries.

La seconda struct caratterizza le entry della bitmap , sono importanti il numero dell’entry e il numero del bit.

Per una corretta esecuzione della BitMap sono state implementate le seguenti funzioni:

**- void BitMap\_init(BitMap\* b, int bitmap\_blocks, int disk\_blocks, int occupation);**

Inizializza una nuova BitMap vuota.

È una funzione void, quindi non avrà nessun valore di ritorno ma inizializzerà tutti i campi della struct a dei valori di default che indicano che il disco è completamente vuoto.

**- BitMapEntryKey BitMap\_blockToIndex(int num);**

Prende come parametro un intero che indica il numero dei blocchi e restituisce un BitMapEntryKey popolata con rispettivamente il primo campo della struct che indica l’entry della BitMap che contiene l’informazione sul blocco e il secondo campo indica lo spiazzamento all’interno del blocco per individuare il bit di cui abbiamo bisogno.

**- int BitMap\_indexToBlock(int entry, uint8\_t bit\_num);**

Funzione elementare che converte un bit in un indirizzo lineare.

**- int BitMap\_get(BitMap\* bmap, int start, int status);**

Funzione che restituisce l'indice del primo bit con stato status nella BitMap bmap e inizia a cercare dalla posizione start in poi. Restituisce -1 se non viene trovato.

**- int BitMap\_set(BitMap\* bmap, int pos, int status);**

Funzione che setta il bit con indice pos a status. Ha come valore di ritorno status in caso di successo.

**- int BitMap\_test(BitMap\* bmap, int pos);**

Funzione che ritorna lo status del bit con indice pos all’interno della BitMap bmap.