

# Elaborato Assembly

Corso di Architettura degli Elaboratori A.A. 2023/2024  
Prof. Franco Fummi, Prof. Michele Lora

Tommi Bimbato VR500751, Antonio Iovine VR504083

17 maggio 2024

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Approccio progettuale . . . . .	1
1.2	Analisi delle specifiche . . . . .	1
1.2.1	Input . . . . .	2
1.2.2	Algoritmi di pianificazione . . . . .	2
1.2.3	Output . . . . .	2
1.3	Funzionamento logico . . . . .	3
1.3.1	Struttura dati . . . . .	3
1.3.2	Schema generale di funzionamento . . . . .	3
1.3.3	Sorting . . . . .	3
1.3.4	Calcolo penali, salvataggio risultato ed uscita . . . . .	3
1.4	Codice . . . . .	3

## **Sommario**

L'obiettivo del progetto è sviluppare un software per la pianificazione delle attività di un sistema produttivo. La produzione è organizzata in slot temporali uniformi, durante i quali un solo prodotto può essere in fase di lavorazione. Il software consentirà di ottimizzare la pianificazione delle attività secondo due algoritmi di pianificazioni differenti. L'intero software è stato sviluppato in linguaggio Assembly e testato su un insieme di dati di prova allegati a questa documentazione.

## Capitolo 1

# Introduzione

### 1.1 Approccio progettuale

L'elaborato è stato condotto seguendo un approccio metodologico strutturato. Inizialmente, è stata eseguita un'analisi dettagliata per identificare i requisiti e le funzionalità principali del software. Questo processo ha consentito una comprensione completa del contesto operativo e degli obiettivi da raggiungere.

Successivamente, è stata sviluppata una bozza del software utilizzando il linguaggio di programmazione C. Questo ha permesso la traduzione dei requisiti in una struttura logica e l'identificazione dell'architettura generale del software.

Parallelamente, sono stati definiti gli spazi di memoria necessari per la memorizzazione dei dati durante l'esecuzione del programma. Ciò ha garantito un utilizzo efficiente delle risorse disponibili.

Successivamente, è stata sviluppata una stesura iniziale del codice assembly, ottimizzando l'efficienza e la velocità di esecuzione del software.

Infine, sono stati eseguiti test approfonditi per verificare il corretto funzionamento del programma e identificare eventuali aree di miglioramento. L'iterazione su questo processo ha portato a modifiche e ottimizzazioni fino al raggiungimento di un livello soddisfacente di prestazioni e funzionalità.

### 1.2 Analisi delle specifiche

La produzione è organizzata in slot temporali uniformi, durante i quali un solo prodotto può essere in fase di lavorazione. Il programma analizza una serie di prodotti, ognuno caratterizzato da un identificativo, una durata, una scadenza e una priorità secondo le specifiche seguenti:

- Identificativo: un codice da 1 a 127;
- Durata: il numero di slot temporali per il completamento (da 1 a 10);
- Scadenza: il limite massimo di tempo entro cui il prodotto deve essere completato (da 1 a 100);

- Priorità: un valore da 1 a 5, che indica sia la priorità che la penalità per il ritardo sulla scadenza<sup>1</sup>.

Per ogni prodotto consegnato in ritardo, l'azienda deve pagare una penale, calcolata moltiplicando la priorità del prodotto per il ritardo in unità di tempo rispetto alla scadenza.

### 1.2.1 Input

Quando l'utente fornisce due file come parametri da linea di comando, il primo viene considerato come l'input, mentre il secondo viene utilizzato per salvare i risultati della pianificazione. Ad esempio:

```
pianificatore Ordini.txt Pianificazione.txt
```

In questo caso, il programma caricherà gli ordini dal file `Ordini.txt` e salverà le statistiche stampate a video nel file `Pianificazione.txt`.

Se l'utente fornisce solo un parametro, la stampa su file verrà ignorata.

Il file degli ordini dovrà avere un prodotto per riga, con tutti i parametri separati da virgola. Ad esempio, se l'ordine fosse:

```
ID: 4; Durata: 10; Scadenza: 12; Priorità: 4;
```

Il file dovrà contenere la seguente riga:

```
4,10,12,4
```

Una volta letto il file, il programma mostrerà il menu principale che chiede all'utente quale algoritmo di pianificazione desidera utilizzare.

### 1.2.2 Algoritmi di pianificazione

Una volta letto il file, il programma visualizzerà il menu principale, permettendo all'utente di selezionare l'algoritmo di pianificazione desiderato. Le opzioni disponibili sono:

1. Earliest Deadline First (EDF): Si pianificano per primi i prodotti con scadenza più vicina. In caso di parità nella scadenza, si considera la priorità più alta.
2. Highest Priority First (HPF): Si pianificano per primi i prodotti con la priorità più alta. In caso di parità di priorità, si considera la scadenza più vicina.

L'utente può selezionare uno dei due algoritmi per la pianificazione delle attività del sistema produttivo.

### 1.2.3 Output

Pippo puppu bau bau ciao ciao come va! hello its me i was wondering if after all these years you'd like to meet to go over everything they say that time's supposed to heal ya but i ain't done much healing...

<sup>1</sup> Il valore 5 indica la priorità più alta.

## 1.3 Funzionamento logico

### 1.3.1 Struttura dati

La lettura del file salva in una variabile buffer alla quale è stata allocata preventivamente 1024k di spazio

### 1.3.2 Schema generale di funzionamento

Acquisizione argomenti -> selezione algoritmo di sorting -> ordinamento -> calcolo penali -> stampa risultati

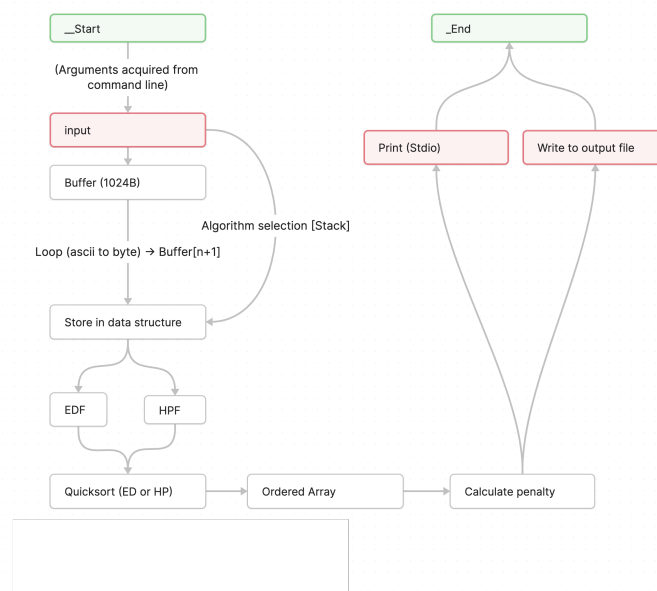


Figura 1.1: Schema generale di funzionamento

### 1.3.3 Sorting

Quick sort bitch!

### 1.3.4 Calcolo penali, salvataggio risultato ed uscita

fprintf

## 1.4 Codice

...

```
34 Codice in ASM!!!
35 int main pippo VOID !!!
36 ; Gibberish Assembly Code
```

```
37 mov eax, 0
38 add ebx, eax
39 sub ecx, ebx
40 mul edx, ecx
41 div esi, edx
42 push eax
43 pop ebx
44 inc ecx
45 dec edx
46 jmp label
47 label:
48 nop
49 cmp eax, ebx
50 jne label
```