UNIVERSITÀ
di VERONA

UNIVR – DIPARTIMENTO DI INFORMATICA

Relazione ASM – Laboratorio Architettura degli Elaboratori

PIANIFICATORE

A.A. 2023/2024

Gruppo di lavoro:

Dalila Portaccio (VR501508)

Louis Opoku (VR486098)

Specifiche

Si sviluppi in **Assembly (sintassi At&t)** un software per la pianificazione delle attività di un sistema produttivo, per i successivi 10 prodotti, nelle successive 100 unità di tempo dette "slot temporali". Il sistema produttivo può produrre prodotti diversi, ma produce un prodotto alla volta. La produzione è suddivisa in slot temporali uniformi, e durante ogni slot temporale solo un prodotto può essere in produzione. Ogni prodotto è caratterizzato da quattro valori interi:

- Identificativo: il codice identificativo del prodotto da produrre. Il codice può andare da 1 a 127;
- Durata: il numero di slot temporali necessari per completare il prodotto. La produzione di ogni prodotto può richiedere 1 a 10 slot temporali;
- Scadenza: il tempo massimo, espresso come numero di unità di tempo entro cui il prodotto dovrà essere completato. La scadenza di ciascun prodotto può avere un valore che va da 1 a 100;
- Priorità: un valore da 1 a 5, dove 1 indica la priorità minima e 5 la priorità massima. Il valore di priorità indica anche la penalità che l'azienda dovrà pagare per ogni unità di tempo necessaria a completare il prodotto oltre la scadenza.

Per ogni prodotto completato in ritardo rispetto alla scadenza indicata, l'azienda dovrà pagare una penale in euro pari al valore della priorità del prodotto completato in ritardo, moltiplicato per il numero di unità di tempo di ritardo rispetto alla sua scadenza. Ad esempio, se il prodotto:

Identificativo: 4; Durata: 10; Scadenza: 25; Priorità: 4;

venisse messo in produzione all'unità di tempo 21, il sistema completerebbe la sua produzione al tempo 30, con 5 unità di tempo di ritardo rispetto alla scadenza richiesta, l'azienda dovrebbe pagare una penalità di 5 * 4 = 20 euro.

Il software dovrà essere eseguito mediante la seguente linea di comando:

```
pianificatore <percorso del file degli ordini>
```

Ad esempio, se il comando dato fosse:

```
pianificatore Ordini.txt
```

il software caricherà gli ordini dal file Ordini.txt.

Il file degli ordini dovrà avere un prodotto per riga, con tutti i parametri separati da virgola. Ad esempio, se gli ordini fossero:

Identificativo: 4; Durata: 10; Scadenza: 12; Priorità: 4;

Identificativo: 12; Durata: 17; Scadenza: 32; Priorità: 5;

Il file dovrebbe contenere le seguenti righe:

```
4,10,12,4
12,7,32,1
```

Ogni file non può contenere più di 10 ordini.

Una volta letto il file, il programma mostrerà il menu principale che chiede all'utente quale algoritmo di pianificazione dovrà usare. L'utente potrà scegliere tra i seguenti due algoritmi di pianificazione:

1. **Earliest Deadline First (EDF)**: si pianificano per primi i prodotti la cui scadenza è più vicina, in caso di parità nella scadenza, si pianifica il prodotto con la priorità più alta.
2. **Highest Priority First (HPF)**: si pianificano per primi i prodotti con priorità più alta, in caso di parità di priorità, si pianifica il prodotto con la scadenza più vicina.

L'utente dovrà inserire il valore 1 per chiedere al software di utilizzare l'algoritmo EDF, ed il valore 2 per chiedere al software di utilizzare l'algoritmo HPF.

Una volta pianificati i task, il software dovrà stampare a video:

1. L'ordine dei prodotti, specificando per ciascun prodotto l'unità di tempo in cui è pianificato l'inizio della produzione del prodotto. Per ogni prodotto, dovrà essere stampata una riga con la seguente sintassi:
   `ID:Inizio`
   Dove ID è l'identificativo del prodotto, ed Inizio è l'unità di tempo in cui inizia la produzione.
2. L'unità di tempo in cui è prevista la conclusione della produzione dell'ultimo prodotto pianificato.
3. La somma di tutte le penalità dovute a ritardi di produzione.

Struttura del programma

Abbiamo utilizzato quattro funzioni in totale, tutti collegati all'interno del file sorgente `pianificatore.s` del quale ci sarà la spiegazione in seguito:

- `EDF.s`
- `HPF.s`
- `itoa.s`
- `read.s`

pianificatore

Questa è la funzione principale e si occupa di collegare tutte i vari sottoprogrammi. Oltre a questo, ha altri compiti tra cui:

- Prendere come parametro un file **.txt** che si trova nella cartella **Ordini** contente la lista dei prodotti.
- Apre il file scelto come parametro.
- Chiama la funzione `read.s` che legge il contenuto del file **.txt**.
- Chiede all'utente di scegliere un algoritmo con il quale ordinare i prodotti (**EDF** o **HPF**) o di uscire. L'utente dovrà digitare su tastiera "1" per EPF, "2" per HPF o "q" per uscire. In caso fosse premuto un altro tasto il programma richiede l'inserimento
- Dopo la scelta il programma chiama, in base alla scelta fatta, una delle due funzioni **EPF** o **HFP** (oppure esce)
- Se si è pigiato "1" o "2" il pianificatore stamperà nel terminale una lista ordinata dei prodotti letti.
- Se si è pigiato "q" il pianificatore procederà a chiudere il file **.txt** aperto, svuotare lo **stack** e infine uscire dal programma.

read

Permette di leggere i prodotti presenti nel file ***.txt*** scelto. Quando questa funzione viene chiamata, salva l'indirizzo del codice a cui deve tornare in un registro, che viene poi reinserito nello **stack** alla fine della funzione. Questo è necessario perché il puntatore dello stack viene modificato continuamente durante l'esecuzione della funzione.

Il file viene analizzato cifra per cifra controllando che per ogni sezione del prodotto (ID, Durata, Scadenza e Priorità) rispettino le condizioni spiegate nella descrizione del progetto. Quando viene trovata una virgola o un Line Feed, il numero viene convertito in intero e inserito nello **stack**. Inoltre, la funzione mantiene un contatore dei valori inseriti.

EDF e HPF

Riordinano i prodotti letti dalla funzione `read.s`

- EDF riordina per scadenza ed in caso di scadenze uguali, la funzione confronta le priorità per vedere quale dei due ha la priorità più alta.
- HPF riordina per priorità ed in caso di priorità uguali, la funzione confrontano le scadenze per vedere quale dei due ha la priorità più vicina (bassa).

itoa

Converte un intero in ASCII per poterlo stampare nel terminale, poiché in assembly si possono stampare solo dati ASCII e perché i dati salvati nello stack sono codificati come interi.

Il pianificatore, inoltre, prima di chiamarla riserva uno spazio nello stack più in alto per evitare di sovrascrivere gli ordini.

Output

Qui di seguito i risultati dei test ottenuti.

Contenuto del file EDF.txt:

```
60,5,43,1
47,10,20,5
103,8,42,4
40,2,71,2
12,7,28,2
92,9,67,5
19,3,15,1
75,4,53,3
89,6,10,3
57,4,84,2
```

Risultato del riordinamento del file EDF.txt secondo l'algoritmo EDF e HPF:

```
Scegli il tipo di algoritmo: digita 1 per EDF o 2 per HPF.
Digita q per terminare il programma
1

Pianificazione EDF:
89:0
19:6
47:9
12:19
103:26
60:34
75:39
92:43
40:52
57:54
Conclusione: 58
Penalty: 0

Scegli il tipo di algoritmo: digita 1 per EDF o 2 per HPF.
Digita q per terminare il programma
2

Pianificazione HPF:
47:0
92:10
103:19
89:27
75:33
12:37
40:44
57:46
19:50
60:53
Conclusione: 58
Penalty: 154
```

Contenuto del file Both.txt:

```
34,5,72,2
87,2,25,4
62,10,67,3
96,7,36,4
109,8,56,3
48,3,12,5
115,8,20,5
23,6,21,4
7,5,46,4
79,8,82,1
```

Risultato del riordinamento del file Both.txt secondo l'algoritmo EDF e HPF:

```
Scegli il tipo di algoritmo: digita 1 per EDF o 2 per HPF.
Digita q per terminare il programma
1

Pianificazione EDF:
48:0
115:3
23:11
87:17
96:19
7:26
109:31
62:39
34:49
79:54
Conclusione: 62
Penalty: 0

Scegli il tipo di algoritmo: digita 1 per EDF o 2 per HPF.
Digita q per terminare il programma
2

Pianificazione HPF:
48:0
115:3
23:11
87:17
96:19
7:26
109:31
62:39
34:49
79:54
Conclusione: 62
Penalty: 0
```

Contenuto del file None.txt:

```
12,6,26,3
13,7,13,1
107,2,45,3
35,9,23,2
111,8,11,2
37,10,21,3
4,5,15,4
89,3,72,1
31,4,31,5
112,1,8,2
```

Risultato del riordinamento del file None.txt secondo l'algoritmo EDF e HPF:

```
Scegli il tipo di algoritmo: digita 1 per EDF o 2 per HPF.
Digita q per terminare il programma
1

Pianificazione EDF:
112:0
111:1
13:9
4:16
37:21
35:31
12:40
31:46
107:50
89:52
Conclusione: 55
Penalty: 267

Scegli il tipo di algoritmo: digita 1 per EDF o 2 per HPF.
Digita q per terminare il programma
2

Pianificazione HPF:
31:0
4:4
37:9
12:19
107:25
112:27
111:28
35:36
13:45
89:52
Conclusione: 55
Penalty: 173
```