UNIVERSITÀ
di VERONA

Dipartimento di Informatica

Elaborato Assembly
Laboratorio di Architettura degli Elaboratori

Sviluppo software per la pianificazione delle attività di un sistema produttivo

A.A. 2023/2024

Lavoro di gruppo:

Antongiulio Donno (VR502818)
Alexandra Iuliana Grasu (VR501064)

INDICE

INTRODUZIONE E DESCRIZIONE DEL PROGETTO

Il software in questione permette di gestire la produzione di al massimo 10 prodotti in un arco temporale di 100 unità di tempo dette "slot temporali".
Ogni prodotto è dotato di 4 parametri:

- **IDENTIFICATIVO**: un numero compreso tra 1 e 127 che permette di identificare un prodotto da produrre;
- **DURATA**: numero che identifica gli slot temporali necessari per il completamento della produzione. A ogni prodotto sono concessi da 1 a 10 slot temporali;
- **SCADENZA**: tempo massimo, espresso in unità di tempo, entro in cui il prodotto dovrà essere completato. La scadenza può assumere un valore compreso tra 1 e 100;
- **PRIORITÀ**: valore da 1 a 5, dove 1 indica la priorità minima e 5 quella massima. Questo valore indica anche la PENALITÀ, ovvero la somma di denaro che l'azienda dovrà pagare per ogni unità di tempo che supera la scadenza data.

  Ad esempio: Identificativo: 4; Durata: 10; Scadenza: 25; Priorità: 4; ipotizziamo che il prodotto venga terminato al tempo 30.

  30-25=5 -> risultano 5 unità di ritardo 5x4=20 Euro

  In questo caso l'azienda dovrà pagare un totale di 20 euro di penalità a causa del ritardo nella produzione dei prodotti.

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

pianificatore <percorso del file degli ordini>

Una volta avviato il software, esso dovrà dare la possibilità all'utente di scegliere un algoritmo di pianificazione.

Le possibilità sono 2:

- **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.
- **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.

Per scegliere l'algoritmo si dovranno inserire i valori 1 per EDF oppure 2 per HPF.

Pianificati i task, si dovrà stampare un output a video che presenti:

- il nome dell'algoritmo scelto dall'utente;
- l'ordine dei prodotti, specificando l'identificativo e l'unità temporale di inizio produzione;
- l'unità di tempo in cui termina la produzione dell'ultimo prodotto pianificato;
- la somma di tutte le penalità dovute ai diversi ritardi.

Per quanto riguarda l'uscita dal software si potrà scegliere tra l'inserire una voce apposita o ctrl-c.

SVILUPPO DEL SOFTWARE ATTRAVERSO LA SINTASSI AT&T

Partendo dall'etichetta *_start* andiamo ad eseguire 3 *pop* per arrivare ad avere il nome del file nel registro *esi*: abbiamo inserito queste istruzioni poiché quando eseguiamo un programma mediante linea di comando vengono caricati nella pila il numero di elementi digitati e ognuno di essi.

Successivamente è presente una system call che ci permette di aprire il file e, se non si verificano errori, possiamo eseguire una jump all'etichetta *_read* dove possiamo leggere uno ad uno i numeri all'interno del file. Dato che non si riesce a leggere un numero intero ma solo una cifra alla volta, abbiamo creato un algoritmo (*_trasforma*) che ci permette di trasformare la cifra da codice ascii in valore intero e di unirlo in un unico elemento.Inoltre questo algoritmo permette di verificare se è stata letta una virgola:se è cosi' carica il numero creato in precedenza nello stack grazie all'algoritmo *_carica*, se questa condizione è falsa si continua a trasformare la cifra selezionata.

Una volta arrivati alla fine del file e dopo aver caricato tutti gli elementi nello stack, possiamo chiudere il file (*_close_file*) tramite la system call apposita e chiedere all'utente quale dei due algoritmi vuole scegliere: in *_scegli* è presente una system call che permette di stampare a video il menù che mostra all'utente le possibilità di scelta e un'altra system call che permette di inserire il numero dell'algoritmo scelto oppure di uscire dal programma (*exit*). Poiché la cifra è codificata in ascii, abbiamo creato un algoritmo (*_converti*) che trasforma una stringa in un numero.

Dopo averlo convertito, eseguiamo una jump a *_compara* che, con una serie di compare(*cmp*), controlla se eseguire la funzione *primo*,la funzione *secondo*,uscire dal menù o richiedere il numero in caso di inserimento di un valore diverso rispetto a quelli prestabiliti.

Le funzioni *primo* e *secondo* vengono invocate tramite l'istruzione *call* e dunque viene salvato nello stack l'indirizzo dell'istruzione successiva alla *call*. Una volta eseguiti gli algoritmi all'interno della funzione e scaricata la pila fino ad arrivare al valore inserito precedentemente, possiamo eseguire l'istruzione *ret*, la quale prende l'indirizzo e fa saltare il processore a tale istruzione.

Per quanto riguarda la stampa a video dei risultati, poiché non si possono stampare degli interi ma solo stringhe,abbiamo creato degli algoritmi che ci permettono di convertire i valori in stringhe.

Infine ricompare il menù affinché l'utente possa scegliere uno dei due algoritmi da usare oppure uscire dal programma.

SCELTE PROGETTUALI

All'interno del nostro progetto abbiamo deciso di inserire dei messaggi di errore nelle seguenti occasioni:

- Se nella linea di comando viene inserito un nome di un file inesistente,allora viene stampato un messaggio a schermo e si esce automaticamente dal programma;

- Se nel menù l'utente digita, da tastiera,un numero diverso da quelli indicati,allora viene stampato un messaggio a schermo e permette il reinserimento di una nuova cifra.

Per quanto riguarda le due funzioni il concetto alla base è lo stesso ma, dato che si occupano di svolgere due compiti diversi, presentano delle leggere differenze.

Prendiamo come esempio la funzione *primo*:

Inizialmente, abbiamo settato un contatore a 100 poiché è il valore massimo della scadenza che un prodotto può assumere. Questo numero viene decrementato ogni volta che non troviamo un prodotto che presenta lo stesso valore della scadenza . Se invece i due valori coincidono, salviamo il valore della priorità del prodotto in un registro perchè nel caso in cui ci sia un prodotto con lo stessa scadenza, dobbiamo andare a vedere quale dei due ha la priorità più bassa dato che quest'ultimo,in base alle caratteristiche del progetto,ha importanza minore. Una volta arrivato alla fine dello stack (cioè al valore zero) ci sono due soluzioni:

- Se non abbiamo trovato un prodotto con una scadenza che coincide con il valore del contatore, quest'ultimo viene decrementato;
- Se invece lo troviamo,scorriamo di nuovo lo stack dall'inizio,eseguiamo una *pop* per ogni valore del prodotto individuato,inizializziamo i valori selezionati all'interno dello stack a 128 in modo tale che non vengano riconosciuti quando scorriamo ed eseguiamo una *push* sulla cima della pila.

Una volta caricate le caratteristiche del prodotto selezionato, in una etichetta salviamo quante posizioni abbiamo inserito nello stack a partire dal valore caricato dalla *call* in modo tale che quando sarà il momento di introdurre un nuovo valore, sappiamo di quante celle ci dobbiamo spostare per raggiungere la cima della pila.

Infine, dopo aver caricato tutti i valori, stampiamo a video l'algoritmo selezionato dall'utente, l'identificativo di ogni prodotto con a fianco l'unità di tempo in cui è iniziata la produzione, il tempo totale impiegato per realizzare i 10 prodotti e la penalità accumulata.

Per la funzione *secondo* il ragionamento è lo stesso ma cambiano alcune caratteristiche:

In primo luogo, abbiamo settato un contatore a 1 poiché è il valore minimo della priorità che un prodotto può assumere. Questo numero viene incrementato ogni volta che non troviamo un prodotto con la stessa priorità. Se invece i due valori coincidono, salviamo il valore della scadenza del prodotto in un registro perchè nel caso in cui ci sia un prodotto con lo stessa priorità, dobbiamo andare a vedere quale dei due ha la scadenza più alta dato che quest'ultimo,in base alle caratteristiche del progetto,ha importanza minore. Una volta arrivato alla fine dello stack (cioè al valore zero) ci sono due soluzioni:

- Se non abbiamo trovato un prodotto con una priorità che coincide con il valore del mio contatore, quest'ultimo viene incrementato;
- Se invece lo troviamo,scorriamo di nuovo lo stack dall'inizio,eseguiamo una *pop* per ogni valore del prodotto individuato,inizializziamo i valori selezionati all'interno dello stack a 128 in modo tale che non vengano riconosciuti quando scorriamo e eseguiamo una *push* sulla cima della pila.

Una volta caricate le caratteristiche del prodotto selezionato , in una etichetta salviamo quante posizioni abbiamo inserito nello stack a partire dal valore caricato dalla call in modo tale che quando sarà il momento di introdurre un nuovo valore, sappiamo di quante celle di memoria ci dobbiamo spostare per raggiungere la cima della pila.

Infine, dopo aver caricato tutti i valori, stampiamo a video l'algoritmo selezionato dall'utente,l'identificativo di ogni prodotto con a fianco l'unità di tempo in cui è iniziata la produzione, il tempo totale impiegato per realizzare i 10 prodotti e la penalità accumulata.

CONCLUSIONE

Abbiamo creato 3 file con determinate caratteristiche che ci permettono di verificare l'efficacia del nostro programma:

- Un file chiamato EDF.txt che presenta penalità uguale a zero con EDF e maggiore di zero con HPF;
- Un file chiamato Both.txt che presenta penalità uguale a zero con entrambi gli algoritmi;
- Un file chiamato None.txt che presenta penalità maggiore di zero con entrambi gli algoritmi.

Inserendo tali file nel nostro programma abbiamo ottenuto questi risultati:

- EDF.txt

```
Selezione uno dei due metodi
1-EDF
2-HPF
3-ESCI
1

Pianificazione EDF:
100:0
65:2
47:11
102:19
37:22
86:26
29:31
79:32
95:39
17:49
Conclusione: 51
Penalty: 0
```

```
Selezione uno dei due metodi
1-EDF
2-HPF
3-ESCI
2

Pianificazione HPF:
102:0
37:3
100:7
79:9
29:16
17:17
47:19
86:27
65:32
95:41
Conclusione: 51
Penalty: 54
```

- None.txt

```
Selezione uno dei due metodi
1-EDF
2-HPF
3-ESCI
1

Pianificazione EDF:
11:0
103:6
90:8
14:9
7:16
94:23
9:25
123:34
10:38
120:46
Conclusione: 49
Penalty: 8
```

```
Selezione uno dei due metodi
1-EDF
2-HPF
3-ESCI
2

Pianificazione HPF:
103:0
90:2
11:3
94:9
9:11
10:20
123:28
120:32
14:35
7:42
Conclusione: 49
Penalty: 60
```

- Both.txt

```
Selezione uno dei due metodi
1-EDF
2-HPF
3-ESCI
1

Pianificazione EDF:
61:0
16:4
59:5
123:11
113:18
8:20
28:28
90:31
46:36
32:46
Conclusione: 55
Penalty: 0
```

```
Selezione uno dei due metodi
1-EDF
2-HPF
3-ESCI
2

Pianificazione HPF:
61:0
59:4
16:10
123:11
113:18
90:20
8:25
28:33
46:36
32:46
Conclusione: 55
Penalty: 0
```