Laboratorio di Architettura degli Elaboratori

Elaborato Assembly

A.A. 2023/2024

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.

Dunque, nell'esempio precedente, se si utilizzasse l'algoritmo EDF, l'output atteso sarà:

```
Pianificazione EDF:
4:0
12:10
Conclusione: 17
Penalty: 0
```

Mentre, se si fosse usato l'algoritmo HPF:

```
Pianificazione HPF:
12:0
4:17
Conclusione: 17
Penalty: 20
```

In quanto nel primo caso non ci sarebbero penalità, mentre nel secondo caso il prodotto con ID 4 terminerebbe con 5 unità di tempo di ritardo, da moltiplicare per il valore di priorità 4.

L'output del programma dovrà avere la sintassi riportata sopra.

Una volta stampate a video le statistiche, il programma tornerà al menù iniziale in cui chiede all'utente se vuole pianificare la produzione utilizzando uno dei due algoritmi.

L'uscita dal programma potrà essere gestita in due modi: si può scegliere di inserire una voce apposita (*esci*) nel menu principale, oppure affidarsi alla combinazione di tasti *ctrl-C*. In entrambi i casi però, tutti i file utilizzati dovranno risultare chiusi al termine del programma.

Bonus (facoltativo): se l'utente inserisce due file come parametri da linea di comando, il file specificato come secondo parametro verrà utilizzato per salvare i risultati della pianificazione, indicando l'algoritmo usato. Ad esempio:

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

Il programma carica gli ordini dal file Ordini.txt e salva le statistiche stampate a video nel file Pianificazione.txt.

Nel caso l'utente inserisca un solo parametro, la stampa su file sarà ignorata.

Materiale da consegnare

La struttura della cartella dovrà essere la seguente, pena la non ammissibilità dell'elaborato:

- `VRXXXXXX_VRXXXXXX/`
  - `src/`
  - `obj/ (vuota)`
  - `bin/ (vuota)`
  - `Makefile`
  - `Ordini/`
    - `EDF.txt`
    - `HPF.txt`
    - `Both.txt`
    - `None.txt`
  - `Relazione.pdf`

La cartella src dovrà contenere tutto il codice sorgente (assembly), il Makefile dovrà creare i file oggetto in obj/ ed il binario che verrà salvato nella cartella bin. La cartella Ordini/ dovrà contenere almeno quattro file degli ordini diversi, tali che:

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

La cartella Ordini, oltre a questi file, potrà contenere tutti i file che si ritengono necessari per un test esaustivo del sistema.

La relazione dovrà descrivere la struttura del codice (le strutture dati, le funzioni utilizzate, i file in cui è organizzato). Per le parti principali del programma, dovrà descrivere la logica del codice assembly prodotto. Infine, dovrà descrivere i test effettuati per validare la correttezza del software.

Modalità di consegna:

Tutto il materiale va consegnato elettronicamente tramite procedura guidata sul sito Moodle del corso. Sarà attivata un'apposita sezione denominata "Consegna ASM - <mese> <anno>".

Accedendo alla pagina sarà possibile effettuare l'upload del materiale. La consegna dovrà essere effettuata da solamente uno dei due studenti del gruppo, e la consegna del materiale comporta automaticamente l'iscrizione all'appello orale di entrambi gli studenti.

Il codice e la relazione vanno compressi in un unico file tarball denominato:

`VRXXXXXX_VRXXXXXX.tar.gz`

Dove VRXXXXXX rappresenta le matricole degli studenti che compongono il gruppo. Ogni gruppo deve essere formato da **esattamente 2 studenti**. Non è possibile consegnare il progetto individualmente, e non è possibile consegnare il progetto in più di due studenti. Nel caso ci fossero difficoltà nel formare la coppia per lo svolgimento del progetto, su Moodle è disponibile un forum apposito.

Il pacchetto tarball deve contenere un'unica cartella denominata `VRXXXXXX_VRXXXXXX`, organizzata come descritto sopra.

Verranno accettati solo i progetti compressi in formato tarball (.tar.gz oppure .tgz).

Per ottenere il pacchetto come richiesto:

1. Rinominare la cartella contenente tutto il materiale con il nome `VRXXXXXX_VRXXXXXX`.
2. Salire di un livello rispetto alla cartella e lanciare il comando:
   `$> tar cvfz VRXXXXXX_VRXXXXXX.tar.gz VRXXXXXX_VRXXXXXX/`

Ad esempio, se il gruppo è formato da due studenti con matricole VR123456 e VR654321, bisogna ottenere il file `VR123456_VR654321.tar.gz`, contenente la cartella `VR123456_VR654321`, mediante il comando: `tar czfv VR123456_VR654321.tar.gz VR123456_VR654321/`

Scadenza: **23 Giugno 2024**, Ore 23:59:59

Periodo esami orali (indicativo): 25 Giugno – 5 Luglio.

Note importanti:

1. È possibile effettuare più sottomissioni, ma ogni nuova sottomissione cancella quella precedente.
2. Un solo membro del gruppo deve effettuare la sottomissione.
3. Tutti i componenti del gruppo devono essere iscritti alla pagina Moodle del corso.
4. I gruppi possono essere composti da studenti appartenenti ad entrambe le classi, e a tutti gli anni di corso, a patto che tutti i membri del gruppo debbano sostenere l'esame, ossia che nessuno dei membri abbia già verbalizzato sul libretto l'esame di Architettura degli Elaboratori.
5. Non si accettano progetti consegnati via email e/o dopo la scadenza.
6. I progetti che non soddisfano i requisiti sopraelencati non verranno ammessi all'orale e non verranno valutati.
7. **Tutti i progetti verranno testati automaticamente. Solo i progetti che superano i test saranno ammessi alla discussione orale. Per valutare l'ammissione all'orale, verranno valutate**:
   a. La struttura dell'archivio consegnato, e delle cartelle contenute (check automatico);
   b. La relazione, e la presenza di tutte le informazioni minime richieste (semi-automatico);
   c. La correttezza del codice Assembly (automatico);
8. I progetti non ammessi potranno essere visionati e discussi al termine delle sessione su richiesta degli studenti.
9. Qualsiasi chiarimento dovrà essere richiesto attraverso il forum "*Chiarimenti sugli elaborati*" disponibile su Moodle, seguendo le regole descritte nella pagina principale del forum.
10. **Chiunque venga scoperto a consegnare progetti contenenti pezzi di codice assembly generati automaticamente, utilizzando GCC oppure strumenti di AI generativa (e.g., ChatGPT, Gemini, etc.) perderà la possibilità di svolgere l'esame orale nella sessione in cui il progetto viene consegnato.**