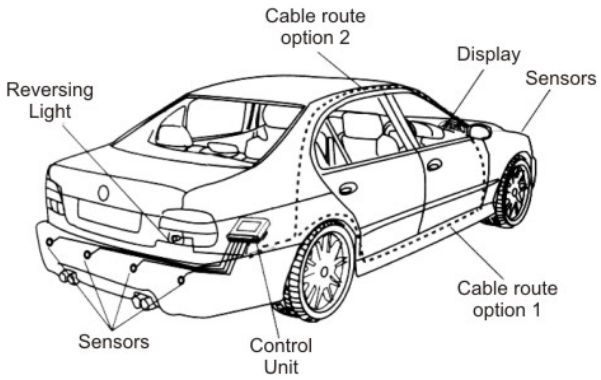
Progetto Assembly MIPS per il Corso di Architetture degli Elaboratori – A.A. 2017/2018 –

Monitoraggio di Sensori in Smart Veichles

Descrizione del Contesto Applicativo

Un'autovettura autonoma è un veicolo automatico in grado di rilevare l'ambiente e la navigazione senza intervento umano. I veicoli autonomi scandagliano l'ambiente con tecniche come radar, GPS, e visione artificiale tramite telecamere. Sistemi di controllo avanzati interpretano le informazioni ricevute per individuare percorsi appropriati, ostacoli e segnaletica rilevante. [*Fonte: Wikipedia*]



Tali veicoli, quindi, si basano *sull'analisi di dati che vengono ricavati periodicamente da sensori predefiniti*, installati in varie sezioni dell'auto. Questi valori vengono *costantemente monitorati*, in modo tale da stabilire se il veicolo sta funzionando nel modo giusto oppure no. Ogni sensore viene installato dal produttore, e *dati relativi al funzionamento atteso vengono forniti* assieme alla documentazione (es. il sensore di rilevazione di ostacoli non può segnalare una distanza negativa dall'ostacolo più vicino). I dati provenienti dai sensori vengono utilizzati dalla centralina di controllo del veicolo per l'esecuzione di processi di ottimizzazione del consumo, stabilizzazione e bilanciamento della marcia, comfort per il guidatore.

I sensori, però, possono incorrere in malfunzionamenti, che portano ad esempio a misurare valori errati, a misurazioni incomplete, o ad una scarsa sincronizzazione con le altre componenti del veicolo.

Descrizione Tecnica

Una nota compagnia italiana ha recentemente iniziato a produrre in massa veicoli con un elevato livello di automazione, con un alto numero di sensori che inviano dati ad una sofisticata unità di controllo in grado di fornire funzionalità innovative ed assicurare una esperienza di guida unica al guidatore.

Unità di Monitoraggio

I progettisti sono però preoccupati dagli effetti che possibili malfunzionamenti di alcuni sensori critici potrebbero avere sull'intero sistema di controllo del veicolo. Incaricano quindi la vostra azienda di sviluppare un software che permetta di stabilire se il sistema funziona correttamente o no (variabile booleana $corr_{sis}$)in base a precise specifiche funzionali. Tale software deve essere eseguito su una piccola unità di monitoraggio ed analisi dei dati con archiettura MIPS, cablata sulla scheda madre dell'unità di controllo, e collegata alle linee dati provenienti da alcuni sensori. Nello specifico:

- **Sensore di Pendenza**: restituisce un numero intero con segno, *p(t)*, che indica la pendenza in gradi sessagesimali del veicolo rispetto al piano orizzontale, rilevata dal sensore al tempo *t*. Ad esempio *p(t)=20*

indica che il veicolo sta affrontando una salita di pendenza 20°, mentre *p(t)=-10* indica che il veicolo è in una discesa di pendenza 10°.

- **Sensore di Sterzo**: restituisce un numero intero *s(t)*, con *0 < s(t) < 100*, che indica in gradi sessagesimali l'angolazione dello sterzo rispetto alla scocca dell'auto, rilevata dal sensore al tempo *t*. Ad esempio *s(t)=50* indica che le ruote sono perfettamente direzionate in avanti, per una marcia rettilinea perfetta; valori *0 < s(t) < 50* indicano che la macchina sta curvando a sinistra; mentre valori *50 < s(t) < 100* indica che la macchina sta curvando a destra.
- **Sensore di Distanza da Ostacoli**: restituisce al tempo *t* un valore esadecimale di tre cifre $d(t)=[d_1(t),d_2(t),d_3(t)]$. La prima cifra $d_1(t)$ indica la tipologia di ostacolo, che può essere 'A' ostacolo fisso, 'B' ostacolo mobile. Le altre due cifre $[d_2(t)d_3(t)]_{16}$, interpretate in codifica naturale, indicano la distanza in metri dall'ostacolo. Ad esempio, *d(t) = A1A* indica un ostacolo fisso ($d_1(t)='A'$) a *26* metri dal veicolo($[d_2(t)d_3(t)]_{16} = 1A_{16} = 26_{10}$).

Specifiche di funzionamento

L'unità di monitoraggio legge i dati dei sensori ad intervalli regolari (es. una volta al secondo), e deve valutarne la correttezza in modo indipendente. I progettisti rendono note le seguenti specifiche di funzionamento dei singoli sensori per determinarne il corretto o scorretto funzionamento:

- **Sensore di Pendenza**: *p(t)* deve essere sempre compreso nell'intervallo ***-60 < p(t) < 60***. Se questa condizione è verificata al tempo *t* allora il sensore è considerato come correttamente funzionante (valore di correttezza $corr_p(t)=1$), altrimenti come malfunzionante ($corr_p(t)=0$).
- **Sensore di Sterzo**: il valore di ***0 < s(t) < 100*** letto non può differire di più di 10 gradi rispetto al valore letto dal sensore all'istante precedente, ovvero ***|s(t) – s(t-1)| ≤ 10***. Se questa condizione è verificata al tempo *t* allora il sensore è considerato come correttamente funzionante (valore di correttezza $corr_s(t)=1$), altrimenti come malfunzionante ($corr_s(t)=0$).
- **Sensore di Distanza da Ostacoli**: il sensore ha una **portata massima di 50 metri**. Il sensore sarà reputato come malfunzionante (quindi $corr_d(t)=0$) in questi casi:
  - I valore di distanza rilevato al tempo *t*, ovvero il valore decimale di $[d_2(t)d_3(t)]_{16}$, è superiore a 50 metri (es. *d = B38*, corrispondente a *56* metri di distanza), **oppure**
  - Al tempo *t* viene rilevato un ostacolo (di qualsiasi tipologia) con distanza zero (es. A00, corrispondente a *0* metri di distanza), **oppure**
  - Il sensore trasmette lo **stesso valore di distanza da ostacoli mobili per più di due misurazioni consecutive** (ad esempio se *d(1)=B20*, *d(2)=B20* e *d(3)=B20*, allora $corr_d(1)=1$, $corr_d(2)=1$, e $corr_d(3)=0$).

Ad ogni istante *t*, una volta che dei valori di correttezza $corr_p(t)$, $corr_s(t)$, $corr_d(t)$ sono stati stabiliti per ciascuno dei sensori, si deve stabilire il valore di correttezza dell'**intero sistema** $corr_{sis}$(t) aggregando tali indici singoli secondo le seguenti politiche:

P1. Il sistema viene reputato funzionante all'istante *t* ($corr_{sis}(t)=1$) se tutti e 3 i sensori sono ritenuti funzionanti allo stesso istante (ovvero $corr_p(t)=1$ *AND* $corr_s(t)=1$ *AND* $corr_d(t)=1$). Altrimenti il sistema viene reputato come malfunzionante ($corr_{sis}(t)=0$).
P2. Il sistema viene reputato funzionante all'istante *t* ($corr_{sis}(t)=1$) se almeno 2 sensori sono ritenuti funzionanti allo stesso istante. Altrimenti il sistema viene reputato come malfunzionante ($corr_{sis}(t)=0$).
P3. Il sistema viene reputato funzionante all'istante *t* ($corr_{sis}(t)=1$) se almeno 1 sensore è ritenuto funzionante allo stesso istante. Altrimenti il sistema viene reputato come malfunzionante ($corr_{sis}(t)=0$).

Dettagli del prototipo

Viene chiesta la realizzazione di un prototipo da laboratorio dell'unità di monitoraggio in cui i valori dei sensori sono simulati da 3 files di testo di **input** all'unità, uno per ciascun sensore.

Ciascun file di testo è costituito dalla sequenza dei valori rilevati da uno specifico sensore, **separati da uno spazio**. Ad esempio, il file che simula la sequenza di valori rilevati dal *Sensore di Distanza Ostacoli* potrà avere il seguente contenuto:

A23 B12 B10 B8 A30 A30 C01

dove *d(1)=A23* è il valore del sensore al primo istante di rilevamento (es. *t=1 sec.*); *d(2)=B12* è il valore del sensore al secondo istante di campionamento (es. *t=2 sec.*), etc.

I tre file di testo dovranno contenere lo stesso numero *N* di rilevamenti, con ***N = 100***, e dovranno avere i seguenti nomi:

- "pendenzaIN.txt" ← sequenza dei valori rilevati dal sensore di pendenza;
- "sterzoIN.txt" ← sequenza dei valori rilevati sensore dello sterzo;
- "distanzaIN.txt" ← sequenza dei valori rilevati dal sensore di distanza da ostacoli.

Il risultato del processo di monitoraggio ed analisi dei dati dovrà essere fornito in output dal programma MIPS, producendo i seguenti **output**:

- **tre file di testo**, uno per ogni sensore, contenenti la sequenza di valori 0 o 1 **separati da uno spazio** che corrispondono ai valori di correttezza di ciascun sensore. Ad esempio, il contenuto del file generato per il sensore di distanza da ostacoli corrispondente alla sequenza dei valori di correttezza $corr_d(1)=1$, $corr_d(2)=1$, $corr_d(3)=0$, $corr_d(4)=1$, sarà:

  1 1 0 1

  Questi tre file di testo di output dovranno avere i seguenti nomi:
  - "correttezzaPendenzaOUT.txt" ← sequenza dei valori di correttezza relativi al sensore di pendenza;
  - "correttezzaSterzoOUT.txt" ← sequenza dei valori di correttezza relativi al sensore dello sterzo;
  - "correttezzaDistanzaOUT.txt" ← sequenza dei valori di correttezza relativi al sensore di distanza da ostacoli.

- **tre file di testo**, uno per ogni politica di aggregazione (P1, P2, P3), contenenti la sequenza di valori 0 o 1 **separati da uno spazio** che corrispondono ai valori di correttezza dell'intero sistema ($corr_{sis}(t)$) in base alla politica di aggregazione selezionata. Ad esempio, il contenuto del file generato relativo alla politica P2 corrispondente alla possibile sequenza dei valori di correttezza $corr_{sis}(1)=1$, $corr_{sis}(2)=0$, $corr_{sis}(3)=0$, $corr_{sis}(4)=0$, sarà:

  1 0 0 0

  Questi tre file di testo di output dovranno avere i seguenti nomi:
  - "correttezzaP1.txt" ← sequenza dei valori di correttezza relativi al sensore di pendenza;
  - "correttezzaP2.txt" ← sequenza dei valori di correttezza relativi al sensore dello sterzo;
  - "correttezzaP3.txt" ← sequenza dei valori di correttezza relativi al sensore di distanza da ostacoli.

Consegna

Utilizzando QtSpim, scrivere e provare un programma in assembly MIPS che prenda in input i tre file di testo con la sequenza dei valori rilevati da ciascun sensore, e che produca in output i sei file di testo di correttezza dei sensori e dell'intero sistema rispettando le specifiche fornite.

Inoltre:

a) Determinare i requisiti di memoria dell'unità di monitoraggio, calcolati come la quantità massima di memoria occupata dal codice prodotto e dai relativi dati. Nel calcolare lo spazio occupato dal codice (segmento .text) si consideri la possibile presenza di pseudo-istruzioni. Nel calcolare lo spazio massimo occupato dai dati si considerino i dati statici (segmento .data), quelli dinamici (allocati a run-time con la sbrk, ovviamente solo se utilizzati nell'esercizio) e lo spazio massimo allocato nello stack.

b) Determinare il tempo complessivo di esecuzione dell'intero programma, considerando la somma dei tempi di esecuzione delle singole istruzioni con le seguenti assunzioni:
   - Ciascuna istruzione di lettura dati da memoria (lw, lh, lb) viene eseguita in 800ps ($800x10^{-12}$sec);
   - Ciascuna istruzione di scrittura dati su memoria (sw, sh, sb) viene eseguita in 700ps ($700x10^{-12}$sec);
   - Ciascuna delle istruzioni rimanenti viene eseguita in 500ps ($500x10^{-12}$sec);
   - Escludere dal calcolo il tempo necessario per leggere/scrivere da/su file (syscall per la read/write).

c) Riportare e discutere nella relazione del progetto i risultati relativi all'occupazione massima di memoria e al tempo complessivo di esecuzione del programma, fornendo tutte le evidenze per permettere di verificare i risultati stessi (es. numero di istruzioni per tipologia, screenshot del segmento dati e dello stack, etc). Inoltre discutere nella relazione i seguenti aspetti:
   - Come potreste diminuire i requisiti di memoria richiesti per l'unità di monitoraggio? Potreste stimare di quanto riuscireste a diminuirli?
   - Come potreste diminuire il tempo di esecuzione del vostro programma? Potreste stimare di quanto riuscireste a diminuirlo?
   - Sarebbe possibile diminuire entrambe le quantità? Se si, come? Se no, perchè?

Possibili Estensioni di Progetto: ← *Facoltativo!*

Gli studenti hanno la possibilità di **estedere** la specifica del progetto **introducendo dei sensori aggiuntivi** rispetto ai tre già previsti, ed estendendo di conseguenza le funzionalità dell'unità di monitoraggio. In questo caso:

1. A livello di specifica dell'esercizio (testo del progetto):
   - Descrivere i sensori aggiuntivi considerati.
   - Definire le specifiche del loro corretto funzionamento o malfunzionamento (ovvero definire quando $corr_{nuovo\text{-}sensore}(t)=0$ o $corr_{nuovo\text{-}sensore}(t)=1$).
   - Definire le nuove politiche di aggregazione (da P4 in poi) che tengano conto dei nuovi sensori per determinare la correttezza del sistema ($corr_{sis}(t)$).
2. A livello del programma in assembly MIPS:
   - Considerare altri file di input aggiuntivi – uno per sensore - per simulare la sequenza delle rilevazioni dei nuovi sensori (nessun vincolo sui loro nomi, né sul formato del loro contenuto).
   - Implementare le regole di correttezza $corr_{nuovo\text{-}sensore}(t)$ in base alle specifiche definite al punto 1.

- Implementare le nuove politiche di aggregazione per determinare la correttezza del sistema ($corr_{sis}(t)$) in base alle specifiche definite al punto 1.
- Generare altri file di output aggiuntivi: uno per ciascun sensore, contenente la sequenza di correttezza $corr_{nuovo\text{-}sensore}(t)$, ed un file per ciascuna nuova politica di aggregazione definita al punto 1, contenente la sequenza di correttezza $corr_{sis}(t)$ della nuova politica. Non si pone nessun vincolo sui nomi dei file, né sul formato del loro contenuto.

Importante

Deve trattarsi di una estensione, e come tale **NON deve modificare il funzionamento "di base" del sistema** così come descritto nel testo del progetto. L'estensione NON sarà presa in considerazione nel caso in cui il funzionamento "di base" non sia stato correttamente implementato.

Note Finali Importanti (Leggere con Attenzione):

- Tutti i file di testo (sia di input che di output) devono risiedere nella **stessa cartella** in cui è presente l'eseguibile *QtSpim*.
- Seguire fedelmente **tutte** le specifiche dell'esercizio (incluse quelle relative ai nomi dei file e al formato del loro contenuto).
- Si può assumere che le sequenze dei valori dei sensori contenute nei files di input siano sintatticamente corrette (formato corretto).
- Rendere il codice **modulare** utilizzando ove opportuno **chiamate a procedure** con l'istruzione **jal (jump and link)**, e **rispettando le convenzioni fra procedura chiamante/chiamata**. La modularità del codice ed il rispetto delle convenzioni saranno aspetti **fondamentali** per ottenere una valutazione positiva del progetto.

Modalità di consegna del progetto (Leggere con Attenzione):

- Per sostenere l'esame è necessario consegnare preventivamente il codice e una relazione (in pdf) sul progetto assegnato
    - Ogni gruppo di lavoro dovrà consegnare un'unica relazione
    - Il codice consegnato deve essere funzionante sul simulatore QtSpim, anche se sviluppato e testato con altri simulatori (es, MARS)
- Un archivio contenente il **codice**, i **file di input** utilizzati per verificare il corretto funzionamento del programma, i relativi **file di output**, e la **relazione** dovrà essere caricato sul sito moodle del corso seguendo l'apposito link che verrà reso disponibile alla pagina del corso
    - Non vengono prese in considerazione altre modalità di consegna
- La scadenza esatta della consegna verrà resa nota di volta in volta
- Discussione e valutazione: la discussione degli elaborati avverrà contestualmente all'esame orale e prevede anche domande sull'assembly e su tutti gli argomenti di laboratorio trattati a lezione.

Struttura della relazione (in formato pdf):

- Info su autori e data di consegna
    - gli autori (e loro indirizzo e-mail – preferibilmente quello universitario@stud.unifi.it)
    - la data di consegna
- Descrizione della soluzione adottata, trattando principalmente i seguenti punti:
    - Descrizione ad alto livello dell'algoritmo (in linguaggio naturale, con flow-chart, in pseudo-linguaggio, etc.), delle strutture dati utilizzate (liste, vettori, etc.) e delle procedure utilizzate (argomenti, funzionalità svolte, risultati prodotti)
    - Uso dei registri e memoria (stack, piuttosto che memoria statica o dinamica)
    - Motivazione delle scelte implementative
- Test di corretto funzionamento
    - In questa sezione dovranno essere fornite le evidenze del corretto funzionamento del programma, ad esempio facendo vedere i valori di correttezza dei singoli sensori e dell'intero sistema (in base alle diverse politiche di aggregazione) in corrispondenza di una o più sottosequenze di input
- Discussione sui requisiti di memoria e sui tempi di esecuzione
    - Discussione dei punti a), b), c) della sezione "Consegna" nel testo del Progetto

- Codice MIPS assembly implementato e commentato in modo chiaro ed esauriente. (nota: nella relazione va inserito **TUTTO** il codice)

Codice

- Ricordarsi di inserire anche nel codice gli autori (e loro indirizzo e-mail) e la data di consegna.
- Seguire fedelmente le convenzioni sull'uso della memoria e sulle procedure.
- Commentare il codice in modo significativo (move $s4,$s3 non significa solo che "copio il contenuto del registro $s3 in $s4"....).