

1 Modificare noop.c per far lampeggiare i LED

- 1.1 In ARM-335x 25.4.1.25: il registro GPIO\_CLEARDATAOUT all'offset 0x190 serve a deassertire una linea di GPIO
- 1.2 Prima versione:

```
int c, state = 0;
for(;;) {
    for (c=0; c<100000000; ++c)
        ;
    gpio1[0x190/4] = 0xf << 21;
    gpio1[0x194/4] = state << 21;
    state = (state+1) & 0xf;
}
```

Funziona? NO!

- 1.3 Controllare il listato assembly (in sert.lst)
- 1.4 Aggiungere 'volatile' alla definizione di gpio1
- 1.5 Aggiungere asm("") al ciclo interno
- 1.6 Funziona? SI!
- 1.7 ATTENZIONE! Il programma funziona solo per caso!
  - 1.7.1 Le variabili automatiche sono poste di norma sullo stack, che non e' stato inizializzato. In questo caso sono cosi' poche che il compilatore ha usato solo registri

2 Organizzazione del codice sorgente:

- 2.1 Organizzazione degli header file
  - 2.1.1 Header file principale: beagleboneblack.h
  - 2.1.2 Sub-header file con definizioni per software: comm.h
  - 2.1.3 Sub-header file con definizioni per hardware: bbb-xxx.h
- 2.2 Trasformazione di noop.c in main.c

```
#include "beagleboneblack.h"
volatile int *gpio1 = (int *) 0x4804c000;
volatile int *cm_per = (int *) 0x44e00000;
void init_gpio1(void)
{
    cm_per[0xac/4] = 0x40002;
    gpio1[0x134/4] &= ~(1<<21)|(1<<22)|(1<<23)|(1<<24);
}
void loop_delay(unsigned long d)
{
    unsigned int c;
    for (c=0; c<d; ++c)
        compiler_barrier();
}
void _reset(void)
{
    int state = 1;
    init_gpio1();
    for(;;) {
        loop_delay(100000000);
        gpio1[0x190/4] = 0xf << 21;
        gpio1[0x194/4] = state << 21;
        state = (state+1) & 0xf;
    }
}
```

- 2.3 Funziona? NO! Leggendo sert.lst scopriamo che \_reset() potrebbe non essere all'indirizzo 0x80000000!

3 Modifichiamo sert.lds in modo che '\_reset' sia sempre posto all'inizio della RAM:

```
[...]
SECTIONS
{
    . = mem_start;
    .text : {
        startup.o(.text)
        . = ALIGN(4);
        *(.text)
        . = ALIGN(4);
    } > ram
[...]
```

- 3.1 Cambiamo il nome della funzione in main.c da \_reset() a main()
- 3.2 Definiamo un file assembler startup.S con la funzione \_reset()

```
.text
```

```

| .code 32
| .global _reset
| _reset:
|     b        main
|
+-----+

```

### 3.3 Controllare la posizione di \_reset() in sert.sym

## 4 Inizializzazione dello stack

### 4.1 Modificare sert.lds in modo da definire uno stack dopo la sezione dati non inizializzati:

```

+-----+
|     _bss_end = .;
|     } > ram
|     .stack : {
|         . = ALIGN(4096);
|         stack_end = .;
|         . = . + 4096;
|         stack_top = .;
|     } > ram
| }
+-----+

```

### 4.2 Aggiungere in startup.S la inizializzazione dello stato della CPU:

```

+-----+
|     .text
|     .code 32
| #define SYS_MODE 0x1f
| .global _reset
| _reset:
|     cpsid     if,#SYS_MODE
|     ldr       sp,=stack_top
|     b         main
|
+-----+

```

## 5 Riscrivere loop\_delay() con una barriera di memoria completa

### 5.1 Una barriera di memoria puo' avere diverse funzioni:

- 5.1.1 Impedire che il compilatore ottimizzi e riordini la sequenza delle istruzioni a livello di codice sorgente
  - 5.1.2 Impedire che il processore riordini nella pipeline la sequenza di istruzioni macchina effettivamente eseguite
  - 5.1.3 Impedire che il processore riordini la sequenza di accessi alla memoria
  - 5.1.4 Impedire che il processore permetta l'esecuzione di una istruzione macchina prima del completamento di un accesso alla memoria precedente
  - 5.1.5 Riferimento: "ARM Cortex-A Series Programmer's Guide", 10.2
- 5.2 Aggiungiamo al file bbb\_cpu.h la definizione di macro che realizzano queste barriere:

```

+-----+
| #define compiler_barrier() \
|     __asm__ __volatile__ (" ::: \"memory\")
| #define data_memory_barrier() \
|     __asm__ __volatile__ ("dmb sy" ::: \"memory\")
| #define data_sync_barrier() \
|     __asm__ __volatile__ ("dsb sy" ::: \"memory\")
| #define instr_sync_barrier() \
|     __asm__ __volatile__ ("isb sy" ::: \"memory\")
|
+-----+

```

### 5.3 Riscrivere loop\_delay() in comm.h utilizzando data\_sync\_barrier():

```

+-----+
| static inline
| void loop_delay(unsigned long d)
| {
|     while (d-- > 0)
|         data_sync_barrier();
| }
+-----+

```

#### 5.3.1 Rimuovere la definizione di loop\_delay() in main.c

#### 5.3.2 La nuova primitiva di sincronizzazione rende ogni iterazione del ciclo molto piu' lenta, quindi abbassiamo il numero di cicli eseguiti da 100000000 a 10000000

## 6 Creiamo un file 'init.c' per l'inizializzazione dell'ambiente di esecuzione

### 6.1 Definiamo una funzione \_init() per eseguire le inizializzazioni e saltare poi a main

- 6.1.1 Modificare \_reset in modo che salti a \_init invece di main
- 6.1.2 Aggiungere a comm.h il prototipo di main()
- 6.1.3 Scrivere la funzione \_init():

```

+-----+
| void _init(void)
|
+-----+

```

#### 6.1.4 Scrivere la funzione fill\_bss():

### 6.1.5 Definire il tipo di dati u32 in beagleboneblack.h:

## 7 Definiamo un metodo uniforme per accedere ai registri delle periferiche

7.1.1 E' inefficiente perche' ogni accesso ad un registro di periferica richiede (1) un accesso in RAM per leggere la variabile gpio1, e (2) un accesso alla memoria di I/O della periferica

### 7.2.1 Aggiungiamo in beagleboneblack.h:

### 7.2.2 Creiamo un nuovo file bbb\_gpio.h contenente:

### 7.2.3 Per accedere al registro della periferica:

### 7.3 Spostiamo l'inizializzazione del modulo GPIO1 in init.c:

```
static void init_gpio1(void)
{
    u32 mask = (1 << 21) | (1 << 22) | (1 << 23) | (1 << 24);
    iomem(CM_PER_GPIO1_CLKCTRL) = 0x40002;
    iomem_low(GPIO1_OE, mask);
    iomem_high(GPIO1_IRQSTATUS_CLR_0, mask);
    iomem_high(GPIO1_IRQSTATUS_CLR_1, mask);
}

void _init(void)
{
    init_gpio1();
    fill_bss();
    main();
}
```

+-----+  
7.4 Definire un nuovo header file bbb\_cm.h per il modulo CM\_PER:  
+-----+

```
#define CM_PER          0x44e00000  
#iomemdef(CM_PER_GPI01_CLKCTRL, CM_PER+0xac);
```

+-----+

7.5 Modifichiamo main.c

```
+-----+  
void main(void)  
{  
    int state = 1;  
    for(;;) {  
        loop_delay(10000000);  
        iomem(GPI01_CLEARDATAOUT) = 0xf << 21;  
        iomem(GPI01_SETDATAOUT) = state << 21;  
        state = (state+1) & 0xf;  
    }  
}  
+-----+
```

=====

```
/*  
vim: tabstop=3 softtabstop=4 expandtab list colorcolumn=74  
*/
```