SEP (Sistemi Elettronici Programmabili) → S. Di Pascoli 25/09/17

(esercitazioni a gruppi di 3: mercoledì) → 4/5
↳ esitano la prova pratica all'esame

Ricevimento: venerdì 15:30 - 18:30
stefano.dipascoli@unipi.it
050 - 2217636

Esercitazione 12 pt => in media 10/12 pt
Problema (30 min) 10 pt
Domande orali 10 pt → si può chiedere la domanda di recupero da 5 pt

Preappello => 2 domande scritte (dicembre)

→ Dispense
Datasheet

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

Programma:
- microprocessore ATMEL ATMEGA32
- programmazione in assembler
- programmazione in C

→ ATMEL ATMEGA32 => è un processore tradizionale, simile a tanti altri (non è il più usato, quello è l'ARM)
→ l'ARM viene escluso perché è troppo complesso, a differenza dell'AVR

AVR è compatto
Se l'ATMEGA non è abbastanza potente, si passa all'ARM (o all'XMEGA)

ATMEGA328 → usato in Arduino

Si utilizza la scheda STK500 => contiene un processore ed elettronica generica

Firmware → programma fisso all'interno di un microprocessore
↳ si trovano apparecchiature commerciali programmabili con un sistema operativo real time

FRERTOS, CHIBIOS: sistemi operativi → Re un clock

Nei microcontrollori c'è il firmware → unica parte real time
↳ privo di sistema operativo, viene poi collegato al computer che lavora non real time => hanno un buffer

Sensori → µC → Computer

Le applicazioni strettamente real time sono semplici e quindi gestibili col firmware

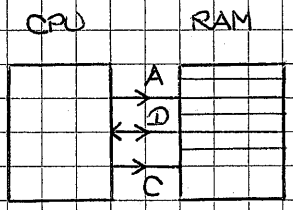
Microcontrollori:

8051
65HJ11
280
PIC → solo ASM, no C

} vecchia generazione, hanno pochi registri
si programmano solo in assembler (ASM, no C)
↓
si potrebbe, ma è sconsigliato

ATMEGA32 → programmabile in C
ATMEL AVR
TEXAS Instruments MSP430
ARM → nato come processore, quindi è programmabile in C
pur essendo dell'82

→ VON NEUMANN = il processore conserva dati e programmi in una memoria



La CPU prende i dati della RAM e li interpreta
→ si può leggere/scrivere una cella per volta

A (address)
D (data, bidirezionale)
C (control)

→ HARVARD = sviluppata per i DSP (digital signal processor)
utilizza 2 memorie (una per i dati, una per le istruzioni)
↳ possono avere tecnologie diverse



(programmabile)
(dati rom)

→ Memorie
non volatili ⇒ non richiedono energia per conservare i dati (es. USB)
volatili ⇒ statiche o dinamiche

per memorie piccole ← per conservare c'è un forzamento hanno bisogno di energia

anche in presenza di energia non conservano il dato
⇒ necessità di refresh (ogni ~4 ms)
↓
dipende dalla temperatura

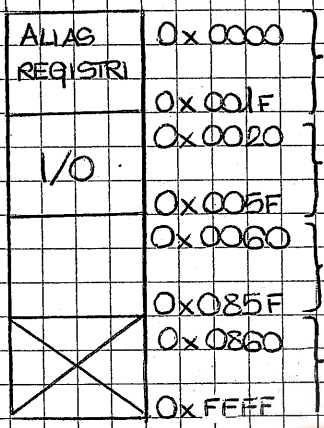
- Non volatili
  - ROM (Read Only Memory)
  - PROM (Programmabile ROM)
  - EPROM (Erasable PROM, cancellabili tramite UV)
  - E^2PROM → cancellabili elettricamente
  - FLASH → simili, ma la cancellazione è globale (nelle USB, costano poco)

- CPU → processore
  Contiene:
  - ALU
  - Decodificatore di istruzioni
  - Registri (da R0 a R31 anziché usare indirizzi)
- Memoria dati → a 8 bit (16 bit per byte, 2 celle appaiate)
  ↓ 0 ; 255 o −128 ; 127

  RAM ⇒ 16 bit ⇒ 2^{16} (65536) celle possibili (non sono tutte implementate)
  → indirizzi

  0x ⇒ esadecimale

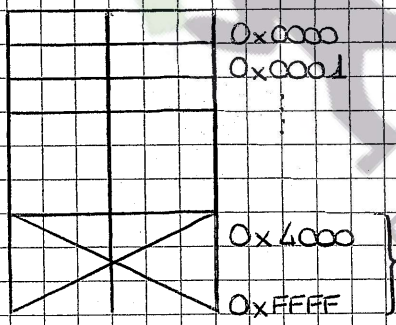
MASTER COPY
Tel. 050 8312126
Cell. 388 9837745



(Mappa della memoria)

Long word = 32 bit
short word = 16 bit

- FLASH → indirizzi a 16 bit (word address)
  istruzioni a 16 bit ⇒ normalmente non si accede a ½ cella



65536 → 64 K

64K x 16

128K x 8

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

Alla partenza di Ra il RESET

PC (program counter, 16 bit, dentro il processore, tiene il conto dell'istruzione) → preFetch

MOV r9, r2 (mette r2 in r9, il vecchio r9 è cancellato)

destinatario sorgente

ADD r9, r4 (per fare l'operazione a 2 operandi A+B → A, A+B → B
non esistono operazioni a 3 operandi)

→ Modi di indirizzamento = servono a specificare l'operando
- immediato (l'operando immediato è una costante)

LDI r17, 37 (mette 37 in r17)
→ può essere solo sorgente, l'operando costante è nell'istruzione

- diretto

LDS r17, 0x0060
↑ indirizzo del dato

STS 0x0060, r17 (può essere usato come destinazione)

→ ISTRUZIONI (aritmetiche & logiche)

- ADD (somma 2 numeri a 8 bit)
- ADC rd, rs rd ← rs + rd + c → può nascere un bit in più (CARRY)
  → c = 0 oppure 1

Es. 0xC8 = 208

```
  1100 1000 +
  1100 1000 =
-------------
c=1 1001 0000
```

Riporto a più cifre → si arriva con 4 addendi (in binario)

→ Presi due # a 16 bit:

0x 12 09
0x 34 E2

è possibile avere un riporto finale

somma le parte meno significative con ADD

somma le parti più significative con ADC

- ADIW → somma a 16 bit con un dato immediato
  il destinatario usa una coppia di registri

  ADIW r4, 0x1234 → opera con rd t.c. d = 24, 26, 28, 30

  r5 : r4 ← r5 : r4 + 0x1234

- SUB (per la sottrazione)
- SBC
- SUBI → sottrazione con dato immediato
- SBCI (non esiste ADDI, usa SUBI r17, -17)
- SBIW → subtract immediate from word
  rd, K

- AND (and bit a bit tra due # a 8 bit) → distruttivo!
- ANDI (and con un dato immediato → UTILE!) ⇒ serve per verificare se un bit è a 1 o a 0 (es vedere se un pulsante è stato premuto)
  ↳ maschera, se ha tutti 0 e un 1 in una certa posizione

```
yyyy yxyy
0000 0100
---------
0000 0x00
```

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

- OR r9, r7
- ORI

- EOR → or esclusivo
- COM → complementa, è a 1 registro (COM r9)
- NEG → cambia segno a un # in C2

- INC incremento } a 1 registro
- DEC decremento

- CLR clear → in realtà non esiste ⇒ CLR r9 in realtà è EOR r9 (ottengo comunque 0)
- SER set (pseudo istruzione)

Gli ATMEGA hanno le moltiplicatore

2 numeri a 8 bit moltiplicati tra loro ne danno uno a 16 bit

- MUL r9, r7 r1 : r0 ← r7 * r9 (SENZA SEGNO)
  (r1 e r0 non possono essere usati come sorgente)

C2 → è trasparente (dati due numeri in C2 sommati come fossero senza segno, ottengo comunque il risultato in C2) → o sottratti

- proprietà non valida per la moltiplicazione
- corretto solo se è rappresentabile (es -200 non va bene)