

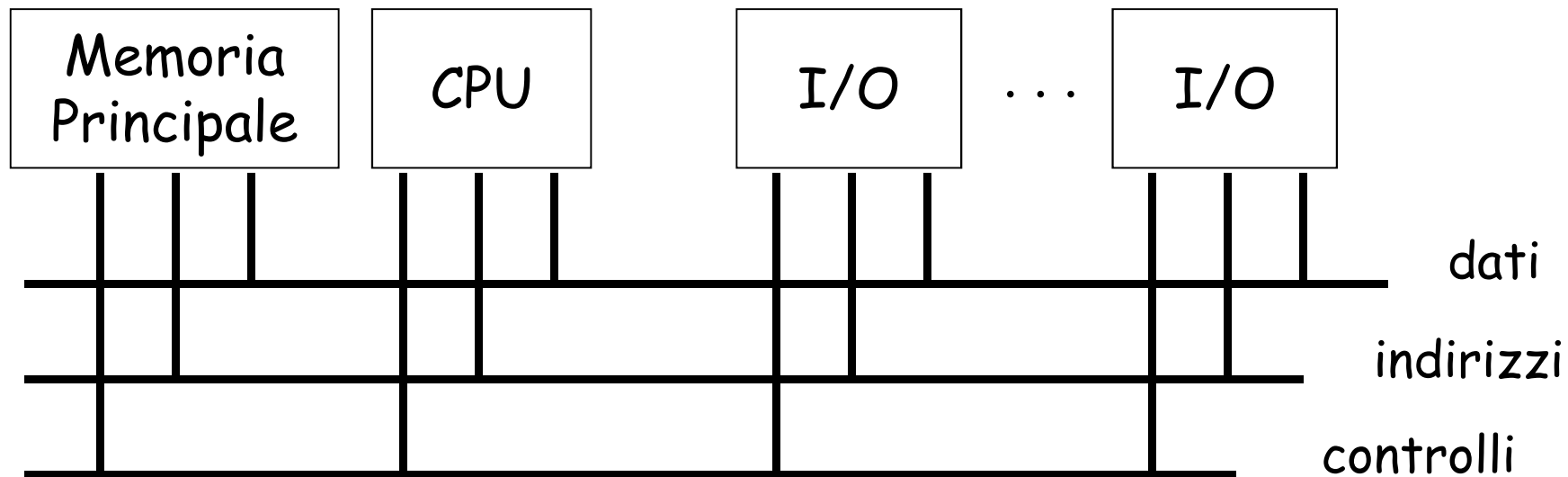


**Fondamenti di Informatica**  
**Ingegneria Clinica**  
**Lezione 05/11/2009**



**Prof. Raffaele Nicolussi**  
**FUB - Fondazione Ugo Bordoni**  
**Via B. Castiglione 59 - 00142 Roma**

# Interazioni tra le unità



Le unità sono connesse attraverso linee di comunicazione specifiche (**bus**) per dati (**data bus**), controlli (**control bus**) e indirizzi (**address bus**)

Le interazioni tra le unità possono avvenire secondo varie modalità

**Modalità elementare:** ogni trasferimento di dati è attivato e controllato esclusivamente da programma

# Sistemi Operativi



Un **sistema operativo** (operating system) è una collezione di programmi per la gestione delle risorse presenti in un calcolatore

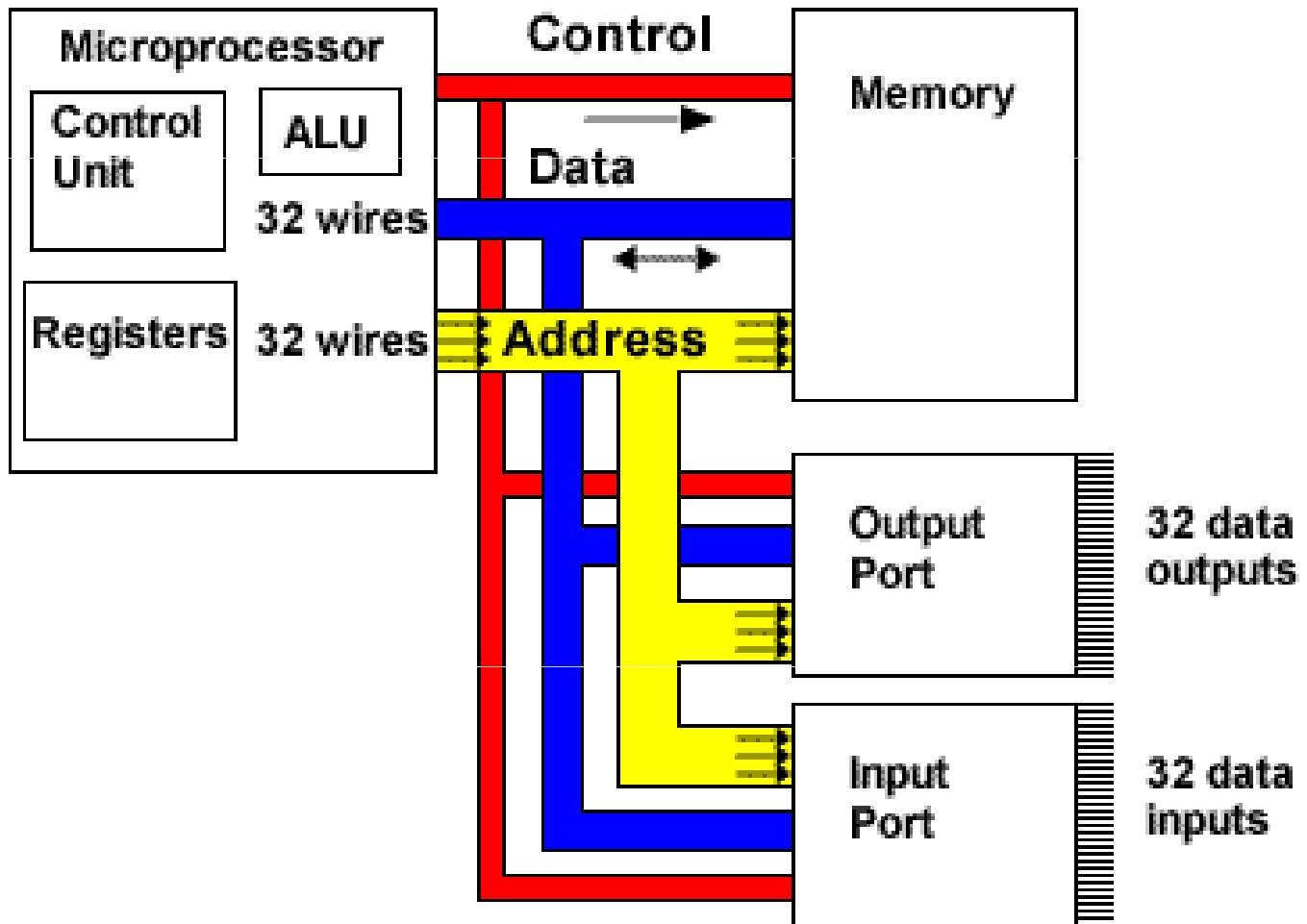
Due obiettivi fondamentali:

- ❑ gestire in modo efficiente l'elaboratore e le sue periferiche
- ❑ creare un ambiente virtuale per facilitare l'interazione uomo macchina

Il sistema operativo mette a disposizione degli altri programmi una serie di funzionalità, semplificando così l'uso del sistema

- ❑ Il codice macchina eseguito dalla CPU si presenta come una successione di porzioni del sistema operativo e porzioni degli altri programmi
- ❑ Per esempio, grazie al sistema operativo, la memoria secondaria appare come una collezione di file opportunamente strutturata, gestibile con semplici comandi
- ❑ Il sistema operativo fornisce un'interfaccia verso l'utente (linguaggio a comandi o interfaccia grafica per l'interazione col sistema)

# Struttura del bus



# Programmazione dei calcolatori



- L'uso del linguaggio macchina è normalmente evitato
- Nella programmazione *reale* si fa uso di **linguaggi assemblativi** e di **linguaggi di alto livello**
- Questi due approcci si distinguono essenzialmente per il livello di astrazione in cui i linguaggi si collocano rispetto alla macchina fisica
- Un tipico **flusso di progetto** di un programma inizia con la sua **scrittura** e finisce con la sua **verifica**
- In tutte le fasi del flusso di progetto, il programmatore può tipicamente contare su un **ambiente/sistema di sviluppo (development environment)** costituito da una opportuna collezione di programmi (e da una macchina fisica su cui eseguirli)

# Algoritmi



- **Definire** un problema significa specificare **i dati** su cui si deve operare ed **il risultato** che si deve produrre
- Con il termine **algoritmo** (metodo risolutivo di un problema) si indica una sequenza ordinata di operazioni, la cui esecuzione nell'ordine specificato produce la soluzione di un problema definito
- La soluzione desiderata deve essere allora fornita in un numero finito di passi per qualsiasi configurazione ammissibile dei dati di input
- L'**efficienza** di un algoritmo è normalmente misurata dalle risorse di tempo e memoria necessarie per la sua esecuzione

# Linguaggi Assemblativi



- Un **linguaggio assemblativo** (**assembly language**), o *assembler*, è un linguaggio di programmazione per una specifica CPU
- Le sue istruzioni corrispondono *direttamente* ad istruzioni macchina della CPU e sono rappresentate da codici *mnemonici* che ne semplificano l'interpretazione
- Esempio: l'istruzione "000000 00101 01111 10000 00000 100000" (istruzione aritmetica: **somma** i contenuti dei registri **5** e **15** e memorizza il risultato nel registro **16**) diventa "**add \$s0, \$a1, \$t7**"
- Un **assemblatore** (**assembler**) è un programma capace di tradurre un programma in assembler (**codice sorgente** (**source code**)) in un corrispondente programma in linguaggio macchina (**codice oggetto** (**object code**))
- L'assemblatore, quindi, ha funzionalità simili al compilatore
- Per poter essere *eseguito* su una determinata macchina, il codice oggetto deve essere trasformato in **codice eseguibile** (**executable code**)



## Esempio di assembler

```
0400 2073FE JSR $FE73      s~
0403 A200   LDX ##0       "□
0405 BD8004 LDA $480,X    =□\
-----
0408 F006   BEQ $410     p✓
040A 2075FE JSR $FE75     u~
040D E8     INX          h
040E D0F5   BNE $405     Pu
0410 00     BRK          □
0411 B9     *=$480
0480 48     'H          H
0481 45     'E          E
-----
0482 4C     'L          L
0483 4C     'L          L
0484 4F     'O          O
0485 00     $0          □
0486 67     !
```





## Esempio CPU didattica : somma di due interi

- Assumiamo che gli interi occupino 2 byte e si trovino in memoria centrale, agli indirizzi 1000 e 1002. Il risultato deve essere memorizzato in 1004
  - Usiamo la “CPU didattica” composta da due registri (R0 ed R1) ed in grado di eseguire la somma usando R0 come accumulatore
  - I registri, come già accennato, sono zone di memoria “particolari” in cui memorizzare dei dati. Sono una sorta di “variabili” della CPU

### □ Codice Assembler

```
mov R0, [1000]  
mov R1, [1002]  
add R0, R1, R0  
mov [1004], R0
```

# Linguaggi di alto livello



- Un **linguaggio di alto livello** (**high level language**) permette l'uso di istruzioni molto potenti (il linguaggio C, presentato nel corso, è di questo tipo)
- E' progettato per essere facilmente utilizzabile dal programmatore, anche per la descrizione di algoritmi molto complessi
- A differenza di un linguaggio assemblativo, un linguaggio di alto livello è notevolmente *lontano* dalla macchina fisica su cui lo si vuole eseguire (programmazione ad alto livello di astrazione)
- (*Un approccio possibile per*) l'esecuzione su una specifica macchina di un programma in un linguaggio di alto livello richiede la *traduzione* di questo *in un* corrispondente programma nel linguaggio macchina specifico (caso tipico per il C)

# Compilatori

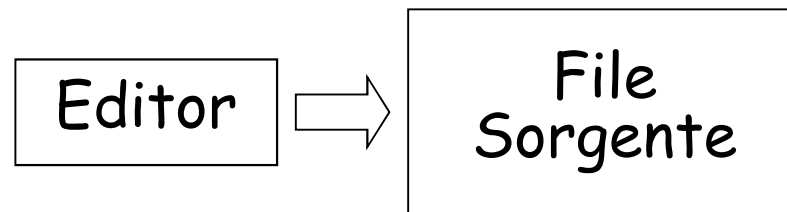


- Un **compilatore** (**compiler**) è un programma capace di tradurre un programma espresso in uno specifico linguaggio di alto livello (**codice sorgente**) in un programma espresso in uno specifico linguaggio macchina (**codice oggetto**)
- Tipicamente, data la potenza dei linguaggi di alto livello, una singola istruzione di un linguaggio di alto livello corrisponde ad una sequenza di molte istruzioni macchina
- La disponibilità di compilare uno stesso linguaggio di alto livello per diversi linguaggi macchina consente al programmatore di scrivere programmi **portabili**
- Per poter essere *eseguito* su una determinata macchina, il codice oggetto deve essere trasformato in **codice eseguibile**

# Progetto: Scrittura



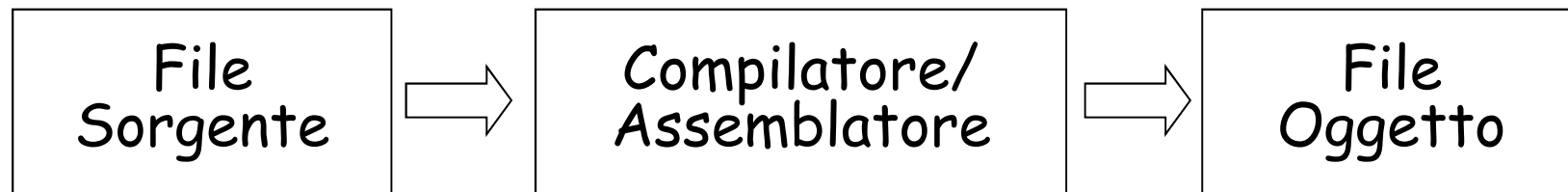
- ❑ Per la generazione di codice sorgente viene tipicamente utilizzato un programma detto **editore** (**editor**)
- ❑ Le funzionalità essenziali di un editor sono quelle richieste nella scrittura di un *semplice* testo (un ambiente di sviluppo contiene tipicamente un editor con funzionalità extra che semplificano la preparazione del testo nel linguaggio specifico)
- ❑ Il testo prodotto viene registrato su memoria secondaria (disco) (generazione del **file sorgente** (**source file**))
- ❑ Il file sorgente è usato come input nella fase successiva



# Progetto: Compilazione



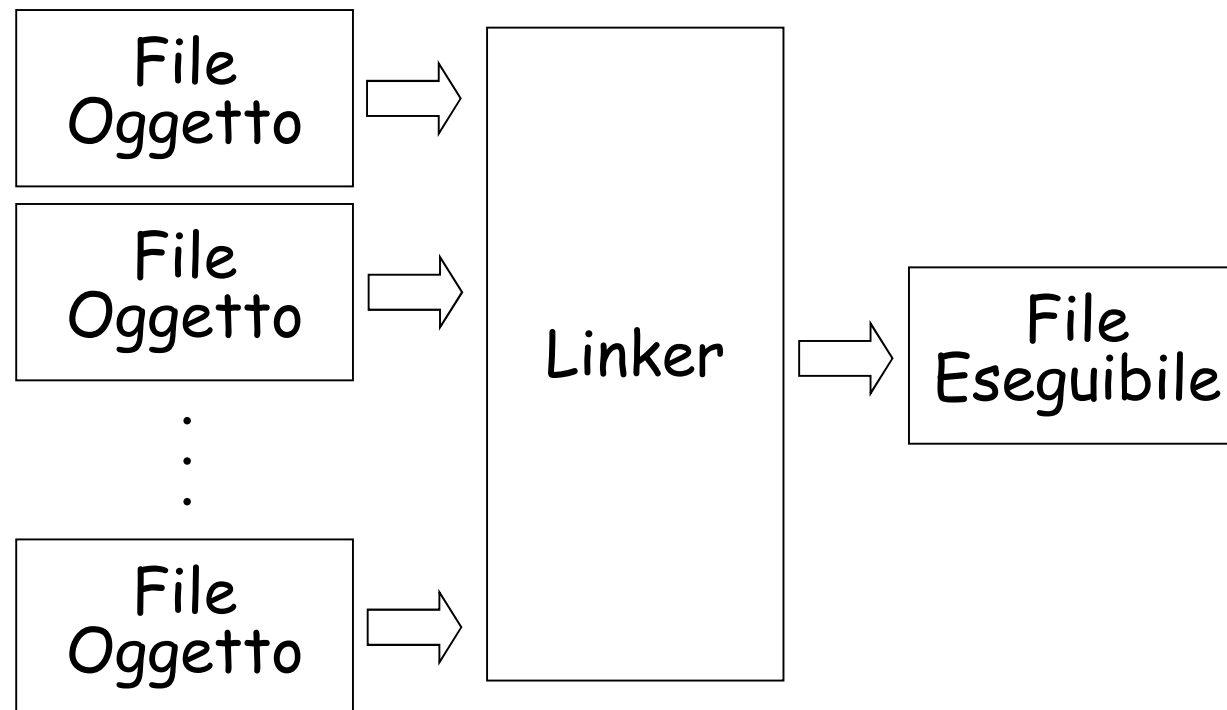
- Il codice contenuto nel file sorgente viene trasformato con un compilatore in codice oggetto (viene generato un **file oggetto** (**object file**))
- Eventuali **errori sintattici** vengono evidenziati (violazioni delle regole di scrittura del codice sorgente)
- Il processo di compilazione prevede l'interpretazione di istruzioni *speciali* presenti nel codice sorgente dette **direttive di compilazione** (in linguaggio C, questo lavoro è a carico di un programma detto **preprocessore** (**preprocessor**), invocato direttamente dal compilatore)
- Il file oggetto è usato come input nella fase successiva



# Progetto: Collegamento



- Il programma finale in linguaggio macchina si compone tipicamente di varie porzioni di codice macchina presenti in più file oggetto
- La generazione del codice eseguibile avviene *collegando* tutti i file oggetto necessari tramite un programma detto **collegatore (linker)** (viene generato un **file eseguibile (executable file)**)



# Esecuzione del programma



- L'esecuzione del programma richiede il *caricamento* nella memoria principale del codice macchina contenuto nel file eseguibile e l'*attivazione* della CPU al prelievo della prima istruzione da eseguire
- Questi compiti sono svolti da un programma detto **caricatore (loader)**
- Il loader è una delle funzionalità offerte dal sistema operativo

