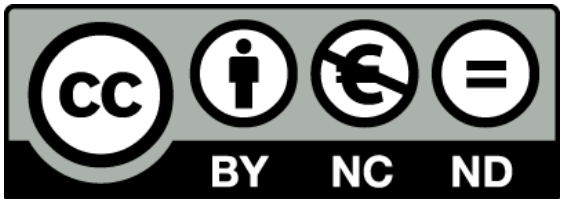
Laboratorio di Programmazione di Sistema

Modi di Indirizzamento

Luca Forlizzi, Ph.D.

Versione 20.1



Introduzione

- Come sappiamo, alcune istruzioni *ASM* ammettono, per alcuni dei loro operandi, diversi modi di indirizzamento
- Esempi
  - in MIPS32:
    `add $t0,$t1,$t3` e `add $t0,$t1,-20`
  - in MC68000:
    `add #100,label` e `add d0,label`
  - in MC68000:
    `move.l memoria,d1` e `move.l d2,memoria`
  - in MC68000:
    `move.l (a0),d3` e `move.l #100,(a0)`
- In questa presentazione approfondiamo il concetto di modo di indirizzamento e descriviamo in dettaglio le più comuni tipologie di modi di indirizzamento definite dagli *ASM-PM*, tra cui alcune mai mostrate in precedenti presentazioni

Indirizzo Generalizzato

- Ad ogni parola, sia di registro che di memoria, è associata una stringa chiamata *indirizzo generalizzato*
- L'indirizzo generalizzato consente di identificare univocamente una parola p, di formato **F**, tra tutte le parole di formato **F**
- Nel caso delle parole di registro standard, l'indirizzo generalizzato è il nome del registro
- Nel caso delle parole di memoria standard, l'indirizzo generalizzato è l'indirizzo di memoria
- Le parole speciali, sia di memoria che di registro, hanno indirizzi generalizzati particolari

Indirizzamento

- Definiamo *indirizzamento* di una parola, l'operazione che consiste in
  1. calcolare l'indirizzo generalizzato della parola
  2. accedere alla parola, in lettura o in scrittura
- Diciamo che *indirizzare* una parola vuol dire effettuare l'indirizzamento di tale parola

Modo di Indirizzamento

- Si definisce *modo di indirizzamento* (*addressing mode*) una regola che, a partire da alcune informazioni, permette di indirizzare una parola
- Ogni volta che si utilizza un'istruzione, per ciascuno degli operandi si deve indicare quale modo di indirizzamento utilizzare
- Tale indicazione viene fornita dalla sintassi dello specificatore di operando (ovvero la porzione della sintassi di un'istruzione relativa ad un certo operando)

Modo di Indirizzamento

- Ogni istruzione, per ciascuno dei suoi operandi, prevede uno o più modi di indirizzamento possibili
- Si possono usare solo i modi di indirizzamento previsti dalla semantica dell'istruzione

Tipici Modi di Indirizzamento per Dati

- Nel seguito, descriviamo le tipologie di modi di indirizzamento che più comunemente vengono usate per accedere ai dati
  - *implicito*
  - *immediato*
  - *diretto-memoria*
  - *diretto-registro*
  - *indiretto-registro*
  - *indiretto-registro con auto-incremento*
  - *indicizzato*
  - *base-indicizzato*
- In future presentazioni, discuteremo altre tipologie di modi di indirizzamento, più frequentemente usate per accedere a istruzioni
- Gli *ASM-PM* di tipo CISC (come M68000, specie nelle versioni più recenti) dispongono di ulteriori modi di indirizzamento, più sofisticati, che non approfondiremo in LPS

Indirizzamento Implicito

- Nei modi di indirizzamento impliciti, l'operando è stabilito dalla semantica dell'istruzione e non può essere modificato dal programmatore
- Per questo non viene (di solito) esplicitato dalla sintassi
- Ad esempio, in MIPS32 l'istruzione `mult A,B` moltiplica i 2 valori a 32 bit contenuti nei registri `A` e `B`; il prodotto è un valore a 64 bit, di cui i 32 bit meno significativi vengono memorizzati in `LO` e i 32 più significativi in `HI`
- Spesso gli operandi impliciti sono registri speciali
- Se un registro general purpose è usato come operando implicito da un'istruzione, acquisisce una funzione speciale
- In alcuni *ASM*, determinate parole di memoria possono essere operandi impliciti

Indirizzamento Immediato

- I modi di indirizzamento immediati permettono di indirizzare una parola che contiene un valore specificato dal programmatore in modo molto efficiente
- Se *i* è un'istruzione che usa un modo di indirizzamento immediato, all'istruzione viene associata una parola di memoria *p*
  - nessuna istruzione diversa da *i* può accedere a *p* con un modo di indirizzamento immediato
  - l'esecuzione di *i* non può modificare il contenuto di *p*, ovvero *i* accede a *p* in sola lettura
  - la sintassi di *i* indica il contenuto di *p*, non l'indirizzo
  - quando si esegue *i*, l'indirizzo di *p* viene calcolato automaticamente e in modo molto efficiente

Indirizzamento Immediato

- In MIPS32, la sintassi di uno specificatore di operando che indica il modo di indirizzamento immediato, prevede semplicemente di scrivere un numero (decimale o esadecimale) o una label
- Esempi in MIPS32-MARS

```
add $t0,$t1,100
li $s0,label
```

- In MC68000, uno specificatore di operando che indica il modo di indirizzamento immediato, è formato dal simbolo # seguito da un numero (decimale o esadecimale) o una label
- Esempi in MC68000

```
add.w #1000,d0
move.l #label,a0
```

Indirizzamento Diretto-Memoria

- Nei modi di indirizzamento diretto-memoria, l'operando è una parola di memoria, il cui indirizzo è indicato in modo esplicito dallo specificatore di operando
- Più esattamente, lo specificatore di operando indica direttamente uno dei seguenti
  - l'intero indirizzo dell'operando
  - una parte dell'indirizzo dell'operando, nel qual caso la rimanente parte dell'indirizzo è specificata in maniera implicita dalla semantica dell'istruzione

Indirizzamento Diretto-Memoria

- In MIPS32-MARS, solo alcune istruzioni possono usare un modo di indirizzamento diretto-memoria
- Lo specificatore di operando che indica un tale modo di indirizzamento è la rappresentazione numerica decimale o esadecimale di un indirizzo, oppure una label
- Si osservi che la sintassi è la stessa usata per l'indirizzamento immediato, ma non vi è ambiguità in quanto nessuna istruzione MIPS32 ha la possibilità di usare, per lo stesso operando, sia il modo di indirizzamento immediato che quello diretto-memoria

Indirizzamento Diretto-Memoria

- Le istruzioni di trasferimento dati, possono usare uno specificatore di operando che indica l'intero indirizzo dell'operando
- Discuteremo altre istruzioni che possono usare un modo di indirizzamento diretto-memoria in future presentazioni
- Esempi in MIPS32-MARS

```
lw $t0,0x1001000C
lh $s0,label
```

Indirizzamento Diretto-Memoria

- MC68000 definisce due modi di indirizzamento diretto-memoria
  - *assoluto lungo*, il quale specifica esplicitamente l'intero indirizzo
  - *assoluto corto*, che specifica in modo esplicito solo una parte dell'indirizzo
- In LPS utilizziamo solo il modo di indirizzamento assoluto lungo, che chiamiamo genericamente indirizzamento diretto-memoria di MC68000
- Lo specificatore di operando che indica un tale modo di indirizzamento è la rappresentazione numerica decimale o esadecimale di un indirizzo, oppure una label

Indirizzamento Diretto-Memoria

- Si osservi quindi che in MC68000 la sintassi dello specificatore di operando consente di distinguere il modo di indirizzamento immediato da quello diretto-memoria
  - nel primo caso lo specificatore è formato da # seguito da un numero o una label
  - nel secondo caso lo specificatore è formato solo da un numero o una label
- La differenziazione sintattica nello specificatore di operando è necessaria, in quanto molte istruzioni possono usare, per lo stesso operando, entrambi i modi di indirizzamento
- Esempi in MC68000-ASM1

```
add.w 1000,d0
add.w #1000,d0
move.l label,a0
move.l #label,a0
```

Indirizzamento Diretto-Registro

- L'operando è un registro, il cui nome è indicato nello specificatore di operando
- In MIPS32 la sintassi dello specificatore di operando prevede di scrivere il nome simbolico oppure l'identificativo numerico del registro, preceduti dal carattere $
- Esempi in MIPS32-MARS

```
lw $t0,0x1001000C
add $s0,$3,$t5
```

Indirizzamento Diretto-Registro

- MC68000 definisce due distinti modi di indirizzamento diretto-registro
  - *diretto-registro dati*, per registri dati
  - *diretto-registro indirizzi*, per registri indirizzi
- Tali modi vengono distinti in quanto alcune istruzioni usano solo uno dei due (ovvero ammettono come operandi solo registri di uno dei due tipi)
- La sintassi dello specificatore di operando prevede di scrivere il nome del registro
- Esempi in MC68000-ASM1

```
add.w d1,d0
move.l d6,a4
```

Indirizzamento Indiretto-Registro

- Nei modi indirizzamento indiretto-registro, l'operando è una parola di memoria, il cui indirizzo è contenuto in un registro
- In altre parole, il registro è un puntatore all'operando
- In MIPS32 la sintassi dello specificatore di operando è (**R**), dove **R** è formato dal carattere $ seguito dal nome simbolico oppure dall'identificativo numerico del registro
- Esempi in MIPS32-MARS

```
lw $t0,($13)
sh $s0,($t1)
```

Indirizzamento Indiretto-Registro

- In MC68000 il registro usato come puntatore deve essere uno degli 8 registri indirizzi
- La sintassi dello specificatore di operando è (**Ax**), dove **Ax** è il nome simbolico di uno dei registri indirizzi
- Esempi in MC68000-ASM1

```
add.b (a2),d0
move.l (a4),label2
```

Indirizzamento Indiretto-Registro con Auto-incremento

- I modi di indirizzamento indiretto-registro con auto-incremento, sono simili ai modi indiretto-registro
- Anche nei primi, l'operando è una parola di memoria e il registro è un puntatore all'operando
- L'indirizzo di tale parola di memoria è pari al contenuto in un registro, eventualmente modificato dall'aggiunta di una costante
- Il contenuto del registro viene modificato dall'aggiunta di una costante (tale modifica permane anche dopo l'esecuzione dell'istruzione)
- In MIPS32 tale modo di indirizzamento non è disponibile

Indirizzamento Indiretto-Registro con Auto-incremento

- In MC68000 esistono due modi di indirizzamento indiretto-registro con auto-incremento:
  - *indiretto-registro con pre-decremento*: il registro viene decrementato di una quantità q e il risultato di tale operazione è l'indirizzo dell'operando
  - *indiretto-registro con post-incremento*: l'indirizzo dell'operando è il contenuto del registro prima dell'esecuzione dell'istruzione; il registro viene incrementato di una quantità q
- In entrambi i casi, la quantità q dipende dal formato specificato dall'istruzione
  - q vale 1 se l'istruzione ha formato byte
  - q vale 2 se l'istruzione ha formato word
  - q vale 4 se l'istruzione ha formato long
- Il registro usato come puntatore deve essere uno degli 8 registri indirizzi

Indirizzamento Indiretto-Registro con Auto-incremento

- Per l'indirizzamento indiretto-registro con pre-decremento, la sintassi dello specificatore di operando è `-(`**Ax**`)`, dove **Ax** è il nome del registro che contiene l'indirizzo
- Per l'indirizzamento indiretto-registro con post-incremento, la sintassi dello specificatore di operando è `(`**Ax**`)+`, dove **Ax** è il nome del registro che contiene l'indirizzo
- Esempi in MC68000-ASM1

```
add.b -(a2),d0
move.l (a4)+,label2
```

Indirizzamento Indicizzato

- Nei modi di indirizzamento indicizzato, l'operando è una parola di memoria, il cui indirizzo è la somma del contenuto in un registro e di un valore costante intero, indicati nello specificatore di operando
- Il registro viene chiamato *registro base* e il valore costante viene chiamato *offset*
- Questo modo di indirizzamento costruisce un puntatore facendo la somma del contenuto del registro base e dell'offset
- Si osservi che il contenuto del registro base non viene modificato
- Il puntatore costruito facendo la somma del contenuto del registro base con l'offset, viene usato per accedere all'operando ma poi viene dimenticato

Indirizzamento Indicizzato

- In MIPS32 la sintassi dello specificatore di operando è **o**(**R**), dove **o** è l'offset, indicato da una label o da una rappresentazione numerica e **R** è il carattere $ seguito dal nome o dall'identificativo numerico del registro base
- MIPS32-MARS consente due modi di indirizzamento indicizzato
  - *indicizzato con offset corto*: l'offset è un intero rappresentabile in complemento a 2 con 16 cifre binarie, che prima di effettuare la somma viene convertito in intero di 32 cifre binarie mediante **sign-extension**
  - *indicizzato con offset lungo*: l'offset è un numero rappresentabile in complemento a 2 con 32 cifre binarie
- In MIPS32-MARS la sintassi dei due modi di indirizzamento indicizzato è la stessa, MARS sceglie automaticamente quale dei due utilizzare in base al valore dell'offset indicato

Indirizzamento Indicizzato

- Esempi in MIPS32-MARS
  ```
  sh $s0,-4($t1)
  lw $t0,label($t0)
  ```
- Il primo esempio usa l'indirizzamento indicizzato con offset corto, il secondo, invece, l'indirizzamento indicizzato con offset lungo

Indirizzamento Indicizzato

- In MC68000 la sintassi dello specificatore di operando è **o**(**Ax**) oppure (**o**,**Ax**), dove **o** è l'offset, indicato da una rappresentazione numerica e **Ax** è il nome del registro base
- Il registro base deve essere uno degli 8 registri indirizzi
- L'offset è un intero rappresentabile in complemento a 2 con 16 cifre binarie, che prima di effettuare la somma viene convertito in intero di 32 cifre binarie mediante **sign-extension**
- Esempi in MC68000-ASM1

```
add.l -4(a2),d0
move.b 10(a4),label2
```

Indirizzamento Base-Indicizzato

- L'operando è una parola di memoria, il cui indirizzo è la somma del contenuto in due registri e di un valore costante intero, indicati nello specificatore di operando
- I due registri vengono chiamati *registro base* e *registro indice* e il valore costante viene chiamato *offset*
- Questo modo di indirizzamento costruisce un puntatore facendo la somma del contenuto dei due registri e dell'offset
- Il contenuto dei due registri non viene modificato
- Il puntatore costruito facendo la somma del contenuto dei registri e dell'offset, viene usato per accedere all'operando ma poi viene dimenticato
- In MIPS32 tale modo di indirizzamento non è disponibile

Indirizzamento Base-Indicizzato

- In MC68000-ASM1 la sintassi dello specificatore di operando è (**o**,**Ax**,**Ry**.**d**) oppure **o**(**Ax**,**Ry**.**d**)
  dove
  - **o** è l'offset, indicato da una rappresentazione numerica
  - **Ax** è uno degli 8 registri indirizzi
  - **Ry** è uno degli 8 registri indirizzi o degli 8 registri dati
  - **d** è una lettera opzionale, detta *dimensione dell'indice*, che può essere `w` oppure `l`
- L'offset è opzionale: se presente è la rappresentazione un intero che appartiene all'intervallo $[-2^7, 2^7)$

Indirizzamento Base-Indicizzato

- L'indirizzo dell'operando si ottiene calcolando la somma dei seguenti addendi (tutti interpretati come numeri con segno):
  - Il contenuto di `Ax`
  - L'offset, se presente, convertito mediante **sign-extension** a una rappresentazione in complemento a 2 formata da 32 cifre binarie
  - Il contenuto della parola di `Ry` di formato `long`, se la dimensione dell'indice è presente e vale `l`
  - Il contenuto della parola di `Ry` di formato `word`, convertito mediante **sign-extension** a una rappresentazione in complemento a 2 formata da 32 cifre binarie, se la dimensione dell'indice non è presente oppure se vale `w`

Indirizzamento Base-Indicizzato

- Esempi in MC68000-ASM1

```
add.w (-4,a2,d5),d0
move.w (10,a4,a3.w),label2
add.w (127,a0,d5.w),d1
move.w (a3,a1.l),d3
```

Indirizzamento Base-Indicizzato

- Nelle versioni più recenti di M68000, a partire da MC68020, il modo di indirizzamento base-indicizzato è potenziato
- Offset, registro base e registro indice sono tutti e tre opzionali (almeno uno, però, deve essere specificato)
- L'offset può essere la rappresentazione di un intero che appartiene a uno tra i seguenti intervalli
  - $[-2^7, 2^7)$
  - $[-2^{16}, 2^{16})$
  - $[-2^{31}, 2^{31})$
- Il contenuto del registro indice, prima di essere sommato agli altri addenti nella costruzione del puntatore, può essere moltiplicato per una piccola costante detta *fattore di scala*

Indirizzamento Base-Indicizzato

- La sintassi dello specificatore di operando è una versione estesa di quella ammessa in MC68000: (**o**,**Ax**,**Ry**.**d**∗**s**) dove
  - **o**, **Ax**, **Ry**, **d** hanno lo stesso significato dei corrispondenti elementi della sintassi MC68000, con la differenza che **o** può qui rappresentare un valore appartenente ad uno degli intervalli di valori più estesi, ammessi in MC68020 e versioni successive
  - **s** è il fattore di scala, ovvero la rappresentazione numerica di una costante intera che può valere 1, 2, 4 oppure 8

Indirizzamento Base-Indicizzato

- Le estensioni introdotte sono particolarmente utili nella gestione di array
- Esempio: accessi ad elementi di array in MC68020

```
* Arr1 è l'indirizzo di un array di long
* Arr2 è l'indirizzo di un array di word
* d1 (word) è un indice in Arr1
* d2 (long) è un indice in Arr2
move.l #Arr1,a3
move.l (a3,d1.w*4),d0
move.w d0,(Arr2,d2.l*2)
* array di 10 elementi di formato long
Arr1:  dc.l 2,5,8,33,392,0,-4,87,-44
* array di 8 elementi di formato word
Arr2:  dc.w 0,0,0,0,0,0,0,0
```