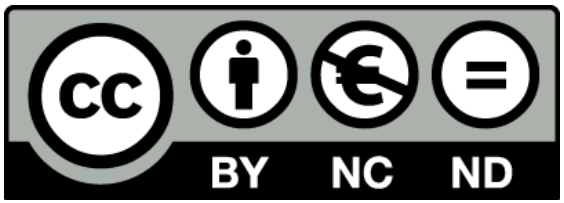
Laboratorio di Programmazione di Sistema

Modi di Indirizzamento

Luca Forlizzi, Ph.D.

Versione 23.1



Introduzione

- Per tradurre in *LM* un qualsiasi programma scritto in un *HLL*, sarebbero sufficienti pochi modi di indirizzamento
- Ad esempio, sarebbe perfettamente adeguata una *ISA* che definisse solo i modi
  - diretto-registro
  - immediato
  - indiretto-registro

Introduzione

- Tuttavia molte *ISA* offrono al programmatore una scelta più ampia di modi di indirizzamento, con un duplice obiettivo
  - facilitare la programmazione rendendo il *LM* più simile ad un *HLL*
  - ottenere maggiore efficienza, in termini di velocità di esecuzione e di occupazione di memoria
- Le *ISA* CISC, in particolare, seguono questa tendenza, in linea con la loro filosofia di "colmare il gap semantico" tra *LM* e *HLL*
- Un *ASM-PM* mette a disposizione i modi di indirizzamento della *ISA* cui è legato e, a volte, modi di indirizzamento aggiuntivi
- Per questo motivo, anche *ASM-PM* di tipo RISC possono presentare un ampio numero di modi di indirizzamento

Introduzione

- In questa presentazione approfondiamo il concetto di modo di indirizzamento in *ASM*, descrivendo in dettaglio le più comuni tipologie di modi di indirizzamento definite da *ASM-PM*, tra cui alcune mai mostrate in precedenti presentazioni
- 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)`

Identificativi

- Ricordiamo che una parola viene indicata da un'istruzione specificando
  - un formato di dato
  - una stringa chiamata *identificativo*
- Se una parola p appartiene al formato **F**, un identificativo di p consente di distinguere univocamente p tra tutte le parole di **F**
- Ovvero ogni parola di **F** diversa da p, ha identificativo diverso dall'identificativo di p
- L'identificativo di una parola di registro standard è il nome del registro
- L'identificativo di una parola di memoria standard è l'indirizzo di memoria
- Altre parole, sia di memoria che di registro, possono avere identificativi differenti

Indirizzamento

- Definiamo *indirizzamento* di una parola, l'operazione che consiste in
  1. determinare l'identificativo della parola
  2. individuare la parola, usando l'identificativo e il formato
  3. accedere alla parola, in lettura o in scrittura
- La parola *indirizzamento* richiama il fatto che per le parole di memoria, che sono la maggior parte, gli identificativi sono indirizzi di memoria
- Diciamo che *indirizzare* una parola vuol dire effettuare l'indirizzamento di tale parola
- Durante l'esecuzione di una istruzione, vengono indirizzati gli operandi dell'istruzione

Modo di Indirizzamento

- Si definisce *modo di indirizzamento* (*addressing mode*) una regola che, a partire da alcune informazioni, permette di indirizzare una parola
- Le regole semantiche di ciascuna istruzione indicano, per ciascuno degli operandi, quali sono i modi di indirizzamento che è possibile utilizzare per tale operando
- In ogni utilizzo di un'istruzione in un programma, la sintassi degli specificatori di operando permette di stabilire quali, tra i modi di indirizzamento possibili, vengono usati per gli operandi
- Nella maggior parte dei casi, tutte le informazioni necessarie per indirizzare un operando sono fornite da un singolo specificatore di operando, ma in alcuni casi (ad es. `btst` in MC68000) più specificatori di operando forniscono informazioni relative ad un singolo operando

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-registro*
  - *diretto-memoria*
  - *indiretto-registro*
  - *indiretto-memoria*
  - *indiretto-registro con modifica automatica*
  - *indiretto con offset*
  - *indiretto con indice*
- In future presentazioni, discuteremo altre tipologie di modi di indirizzamento, più 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, `mult A,B` moltiplica i valori di 32 cifre binarie contenuti dei registri `A` e `B`; il risultato viene memorizzato in `LO` (32 cifre meno significative) e in `HI` (32 cifre più significative), che non sono esplicitati dalla sintassi
- Spesso gli operandi impliciti sono registri speciali
- Se un registro general purpose è usato come operando implicito da un'istruzione, acquisisce una funzione speciale
- Anche determinate parole di memoria possono essere operandi impliciti

Indirizzamento Immediato

- I modi di indirizzamento immediati permettono di indirizzare, in modo particolarmente efficiente, una parola di memoria che contiene un dato specificato dal programmatore
- Se *I* è un'istruzione che usa un indirizzamento immediato, ad *I* viene associata una specifica parola di memoria *P*
  - l'identificativo di *P* è determinato in modo automatico dall'*ASM-PM*, quindi
    - non può essere scelto dal programmatore (ma ciò non è una limitazione)
    - viene calcolato automaticamente e in modo molto efficiente
  - 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

Indirizzamento Immediato

- In MIPS32, la sintassi di uno specificatore di operando che indica il modo di indirizzamento immediato, prevede semplicemente di scrivere
  - un numero in codifica decimale o esadecimale
  - oppure un nome simbolico equivalente ad un numero
  - oppure una label che specifica un indirizzo di memoria
- Esempi in MIPS32-MARS

```
        .eqv    COSTANTE,13
# il terzo operando dell'istruzione seguente usa
# l'indirizzamento immediato espresso come numero
        add     $t0,$t1,100
# il terzo operando dell'istruzione seguente usa
# l'indirizzamento immediato espresso come simbolo
        add     $t2,$t0,COSTANTE
# il secondo operando dell'istruzione seguente usa
# l'indirizzamento immediato espresso come label
        la      $s0,lab1
        ...
lab1:   .word   9
```

Indirizzamento Immediato

- 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
  - un numero in codifica decimale o esadecimale
  - oppure un nome simbolico equivalente ad un numero
  - oppure una label che specifica un indirizzo di memoria
- Esempi in MC68000-ASM1

```
COSTANTE        equ 13
* l'operando sorgente dell'istruzione seguente usa
* l'indirizzamento immediato espresso come numero
        add.w   #100,d0
* l'operando sorgente dell'istruzione seguente usa
* l'indirizzamento immediato espresso come simbolo
        add.w   #COSTANTE,d0
* l'operando sorgente dell'istruzione seguente usa
* l'indirizzamento immediato espresso come label
        move.l  #lab1,a0
        ...
lab1:           dc.w    9
```

Indirizzamento Diretto-Registro

- I modi di indirizzamento diretto-registro permettono di indirizzare una parola di registro
- Se un operando viene indirizzato con l'indirizzamento diretto-registro, viene detto *operando registro*
- Lo specificatore dell'operando indica l'identificativo, ovvero il nome, del registro
- 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

```
# l'istruzione seguente usa per tutti e 3 gli operandi
# l'indirizzamento diretto-registro

# lo specificatore di operando
# - per i primi due è il nome simbolico del registro
# - per il terzo è l'identificativo numerico
        add     $sp,$t1,$10
```

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 considerati distinti in quanto
  - alcune istruzioni usano solo uno dei due (ovvero ammettono come operandi solo registri di uno dei due tipi)
  - il modo diretto-registro indirizzi ha regole particolari in relazione al formato degli operandi
- La sintassi dello specificatore di operando prevede di scrivere il nome del registro
- Esempio in MC68000-ASM1

```
* ind. diretto-registro dati per il primo operando
* ind. diretto-registro indirizzi per il secondo operando
        move.l d6,a4
```

Indirizzamento Diretto-Memoria

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

Indirizzamento Diretto-Memoria

- In MIPS32-MARS
  - solo alcune istruzioni possono usare l'indirizzamento diretto-memoria
  - nell'indirizzamento diretto-memoria, lo specificatore di operando indica esplicitamente l'intero indirizzo di memoria di una parola standard
  - in alcune delle istruzioni che possono usare l'indirizzamento diretto-memoria, discusse in una futura presentazione, vi sono delle limitazioni in merito a quali indirizzi di memoria possono essere usati

Indirizzamento Diretto-Memoria

- Lo specificatore di operando che indica l'indirizzamento diretto-memoria, in MIPS32-MARS può essere
  - una rappresentazione numerica decimale o esadecimale di un indirizzo di memoria
  - una label legata ad un indirizzo di memoria
- 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

- Esempi in MIPS32-MARS

```
# il secondo operando dell'istruzione seguente usa
# l'indirizzamento diretto-memoria espresso come numero
        sh      $t0,0x10001080
# il secondo operando dell'istruzione seguente usa
# l'indirizzamento diretto-memoria espresso come label
        lw      $t2,lab2

# il terzo operando dell'istruzione seguente usa
# l'indirizzamento immediato espresso come numero
        add     $s0,$s1,0x10001080
# il secondo operando dell'istruzione seguente usa
# l'indirizzamento immediato espresso come label
        la      $s0,lab2
        ...
        .data           0x10001000
lab2:   .word   9
```

Indirizzamento Diretto-Memoria

- MC68000 definisce due modi di indirizzamento diretto-memoria
  - *assoluto lungo*, nel quale viene specificato esplicitamente l'intero indirizzo
  - *assoluto corto*, nel quale viene specificato in modo esplicito solo una parte dell'indirizzo, ovvero una stringa di 16 cifre binarie che viene convertita in un indirizzo di 32 cifre binarie mediante un procedimento di *sign-extension*
- 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 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

Indirizzamento Diretto-Memoria

- Esempi in MC68000-ASM1

```
* il primo operando dell'istruzione seguente usa
* l'indirizzamento diretto-memoria espresso come numero
        move.w $4006,d0
* il primo operando dell'istruzione seguente usa
* l'indirizzamento diretto-memoria espresso come label
        move.l lab2,a0

* il primo operando dell'istruzione seguente usa
* l'indirizzamento immediato espresso come numero
        move.w #$4006,d1
* il primo operando dell'istruzione seguente usa
* l'indirizzamento immediato espresso come label
        move.l #lab2,a1
        ...
        org     $4000
lab2:   dc.l    9
        dc.w 0,-1
```

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, l'indirizzo dell'operando è contenuto in un registro
- 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

```
# nelle due istruzioni seguenti, il secondo operando
# usa l'indirizzamento indiretto-registro
        lw $t0,($13)
        sh $s0,($t1)
```

Indirizzamento Indiretto-Registro

- In MC68000 il registro usato dal modo di indirizzamento 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

```
* il primo operando dell'istruzione seguente usa
* l'indirizzamento indiretto-registro
        add.b (a2),d0
* il secondo operando dell'istruzione seguente usa
* l'indirizzamento indiretto-registro
        move.l d1,(a4)
```

Indirizzamento Indiretto-Memoria

- Nei modi indirizzamento indiretto-memoria, l'operando è una parola di memoria, il cui indirizzo è a sua volta contenuto in una parola di memoria
- In altre parole, l'indirizzo dell'operando è il contenuto di una parola di memoria
- MIPS32 e MC68000 non offrono modi di indirizzamento di questo tipo
- Versioni successive di M68000, come MC68020 hanno modi di indirizzamento indiretto-memoria

Indirizzamento Indiretto-Registro con Modifica Automatica

- I modi di indirizzamento indiretto-registro con modifica automatica, sono variazioni dei modi indiretto-registro nei quali
  - Come nei modi indiretto-registro, si indirizza una parola di memoria il cui indirizzo è contenuto in un registro
  - Inoltre, il registro viene modificato dall'aggiunta di una costante
  - L'effetto della modifica permane anche dopo l'esecuzione dell'istruzione
- Questi modi di indirizzamento trovano applicazione in operazioni di scansione di array e di gestione di strutture dati di tipo *stack*
- MIPS32 non offre modi di indirizzamento di questo tipo

Indirizzamento Indiretto-Registro con Modifica Automatica

- In MC68000 esistono due modi di indirizzamento indiretto-registro con modifica automatica:
  - *indiretto-registro con pre-decremento*: il registro viene decrementato di un valore k 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 un valore k
- In entrambi i casi, k dipende dal formato specificato dall'istruzione
  - k vale 1 se l'istruzione ha formato byte
  - k vale 2 se l'istruzione ha formato word
  - k vale 4 se l'istruzione ha formato long
- Il registro usato dal modo di indirizzamento deve essere uno degli 8 registri indirizzi

Indirizzamento Indiretto-Registro con Modifica Automatica

- 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

```
        move.l  #$1004,a2
* l'istruzione seguente somma il contenuto del byte
* di indirizzo $1003 a quello in d0
        add.b -(a2),d0
* al termine dell'esecuzione a2 contiene $1003

        move.l  #$2004,a4
* l'istruzione seguente somma il contenuto della long
* di indirizzo $2004 a quella in d1
        move.l (a4)+,d1
* al termine dell'esecuzione a4 contiene $2008
```

Indirizzamento Indiretto-Registro con Modifica Automatica

- I modi di indirizzamento indiretto-registro con modifica automatica, permettono di realizzare scansioni sequenziali di array in modo semplice, efficiente e simile alle controparti in C

Scansione Array in C

```
int arr[10], *p, sum;
        ...
sum = 0;
p = arr;
while ( p < arr + 10 )
        sum += *p++;
```

Scansione Array in MC68000

```
arr:     dcb.w   10,0
arr_end  equ arr+20
         ...
         clr.w   d0
         move.l  #arr,a0
loop:    cmp.l   #arr_end,a0
         bhs     loop_end
         add.w   (a0)+,d0
         bra     loop
loop_end:
```

Indirizzamento Indiretto con Offset

- I modi di indirizzamento indiretto con offset (detti anche indiretto con spiazzamento) sono variazioni dei modi indiretto-registro o indiretto-memoria nei quali l'indirizzo dell'operando viene calcolato come somma di due termini
  - Un indirizzo, detto *base*, contenuto in un registro o una parola di memoria
  - Un valore costante intero, detto *offset* (o *spiazzamento*)
- La base non viene modificata
- Nella maggior parte dei casi, l'indirizzamento indiretto con offset viene definito nella variante più semplice in cui la base è contenuta in un registro, che viene quindi chiamato (nel contesto del modo di indirizzamento) *registro base*

Indirizzamento Indiretto con Offset

- MIPS32-MARS mette a disposizione due modi di indirizzamento indiretto con offset
  - *indiretto-registro con offset corto*
    - la base è il contenuto di uno dei GPR
    - l'offset è un intero rappresentabile in complemento a 2 con 16 cifre binarie, che viene convertito in intero di 32 cifre binarie mediante **sign-extension** prima di essere sommato alla base
  - *indiretto-registro con offset lungo*
    - la base è il contenuto di uno dei GPR
    - l'offset è un numero rappresentabile in complemento a 2 con 32 cifre binarie

Indirizzamento Indiretto con Offset

- L'indirizzamento indiretto-registro con offset corto ha minore potere espressivo di quello con offset lungo, in quanto qualunque valore dell'offset valido per il primo è valido anche per il secondo, ma non viceversa
- Ma l'indirizzamento indiretto-registro con offset corto ha il vantaggio di essere più efficiente, sia in termini di velocità di esecuzione che di memoria occupata dalla traduzione in *LM*

Indirizzamento Indiretto con Offset

- Per entrambi i modi di indirizzamento indiretto con offset di MIPS32-MARS, 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
- MARS sceglie automaticamente quale dei due modi utilizzare
  - se il valore dell'offset è rappresentabile in complemento a 2 con 16 cifre binarie, MARS usa l'indiretto-registro con offset corto
  - altrimenti usa l'indiretto-registro con offset lungo

Indirizzamento Indiretto con Offset

- Esempi in MIPS32-MARS

```
        la      $t0,0x10000008

# l'istruzione seguente copia una half da t1
# all'indirizzo di memoria 0x10000004 usando
# l'indirizzamento indiretto-registro con offset corto
        sh      $t1,-4(t0)
# al termine dell'esecuzione t0 non è modificato

# l'istruzione seguente copia una half da t2
# all'indirizzo di memoria 0x10000018 usando
# l'indirizzamento indiretto-registro con offset corto
        sh      $t2,0x0010(t0)
# al termine dell'esecuzione t0 non è modificato

# l'istruzione seguente copia una half da t4
# all'indirizzo di memoria 0x10010008 usando
# l'indirizzamento indiretto-registro con offset lungo
        sh      $t4,0x10000(t0)
# al termine dell'esecuzione t0 non è modificato
```

Indirizzamento Indiretto con Offset

- MC68000 ha un modo di indirizzamento indiretto con offset chiamato *indiretto-registro con offset* in cui
  - La base è il contenuto di uno degli 8 registri indirizzi
  - L'offset è un intero rappresentabile in complemento a 2 con 16 cifre binarie, che viene convertito in intero di 32 cifre binarie mediante **sign-extension** prima di essere sommato alla base
- La sintassi MC68000-ASM1 per lo 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

Indirizzamento Indiretto con Offset

- Esempi in MC68000-ASM1

```
        move.l  #$4008,a0

* l'istruzione seguente copia una word da d1
* all'indirizzo di memoria $4004
        move.w  d1,-4(a0)
* al termine dell'esecuzione a0 non è modificato

* l'istruzione seguente copia una word da d2
* all'indirizzo di memoria $4028
        move.w  d2,$0020(a0)
* al termine dell'esecuzione a0 non è modificato
```

Indirizzamento Indiretto con Offset

- L'indirizzamento indiretto con offset è utile, in molti *ASM-PM*, per rendere più efficienti le operazioni di accesso ad elementi di array che soddisfano una delle seguenti condizioni
  - l'indice è un valore noto a tempo di traduzione
  - l'array è allocato staticamente (ovvero è memorizzato in un'area di memoria il cui indirizzo è noto a tempo di traduzione)

Indirizzamento Indiretto con Offset

- Ricordiamo che la traduzione *ASM* di un'operazione di accesso all'elemento di indice k di un array, consiste nel
  - calcolare l'indirizzo dell'area di memoria che contiene l'elemento di indice k, che è pari a

    $$A + k \cdot m$$

    dove A è l'indirizzo dell'area di memoria che contiene l'intero array e m è la quantità di byte usata per memorizzare ciascuno dei singoli elementi dell'array
  - leggere o scrivere il contenuto di tale area di memoria
- Un singolo uso dell'indirizzamento indiretto con offset permette di effettuare sia l'operazione di somma che conclude il calcolo dell'indirizzo, sia l'operazione di lettura o scrittura in memoria

Indirizzamento Indiretto con Offset

- A tale scopo, i due addendi A e $k \cdot m$ vengono rappresentati uno dalla base e uno dall'offset
- Si noti che
  - la base è un indirizzo
  - l'offset è in grado di rappresentare solo addendi il cui valore è noto a tempo di traduzione, quindi è necessario che almeno uno tra A e $k \cdot m$ sia noto a tempo di traduzione
  - A è un indirizzo e quindi deve esse rappresentato da un termine in grado di rappresentare indirizzi
  - A è noto a tempo di traduzione se e solo se l'array è allocato staticamente
  - $k \cdot m$ in gran parte dei casi è un numero non troppo grande
  - nei linguaggi con tipizzazione statica, come il C, m è un valore noto a tempo di traduzione e di conseguenza $k \cdot m$ è noto a tempo di traduzione se e solo se k è una costante

Indirizzamento Indiretto con Offset

- Presentiamo due diverse tecniche che utilizzano un indirizzamento indiretto con offset per tradurre accessi ad array in cui almeno uno tra i termini A e $k \cdot m$ è noto a tempo di traduzione
  - La **Tecnica 1** permette di tradurre accessi ad elementi di array in cui l'indice è un valore noto a tempo di traduzione
    - la base contiene l'indirizzo A dell'array
    - l'offset indica il valore costante $k \cdot m$
  - La **Tecnica 2** permette di tradurre accessi ad elementi di array allocati staticamente (ovvero tali che A è noto a tempo di traduzione)
    - la base contiene il valore $k \cdot m$
    - l'offset indica l'indirizzo A dell'array

Indirizzamento Indiretto con Offset

- Si può applicare la **Tecnica 1** con entrambi i modi di indirizzamento indiretto-registro con offset di MIPS32-MARS
- Se il valore $k \cdot m$ è esprimibile in complemento a 2 con 16 cifre binarie, è conveniente utilizzare l'indirizzamento indiretto-registro con offset corto, che è più efficiente

Accesso Array 1 in C

```
int arr[10];
        ...
// tutti gli indici sono
// valori costanti
arr[2] = arr[5] + arr[1];
```

Accesso Array 1 in MIPS32

```
# rappresenta int con half
# s0 contiene l'indirizzo di
# un'area di memoria in cui
# è rappresentato l'array
lh      $t0,10($s0) # arr[5]
lh      $t1,2($s0) # arr[1]
add     $t1,$t1,$t0
sh      $t1,4($s0) # arr[2]
```

Indirizzamento Indiretto con Offset

- Si può applicare la **Tecnica 1** con il modo di indirizzamento indiretto-registro con offset di MC68000-ASM1 a condizione che il valore $k \cdot m$ sia esprimibile in complemento a 2 con 16 cifre binarie

Accesso Array 1 in C

```
int arr[10];
        ...
// tutti gli indici sono
// valori costanti
arr[2] = arr[5] + arr[1];
```

Accesso Array 1 in MC68000

```
* rappresenta int con word
* a0 contiene l'indirizzo di
* un'area di memoria in cui
* è rappresentato l'array
move.w  10(a0),d1 ; arr[5]
add.w   2(a0),d1  ; arr[1]
move.w  d1,4(a0)  ; arr[2]
```

Indirizzamento Indiretto con Offset

- La **Tecnica 2** può essere applicata solo se l'offset è in grado di memorizzare un indirizzo
- Ciò non accade
  - con il modo di indirizzamento indiretto-registro con offset di MC68000
  - con il modo di indirizzamento indiretto-registro con offset corto di MIPS32
- Infatti in entrambi i casi
  - gli indirizzi di memoria hanno 32 cifre binarie
  - gli offset hanno solo 16 cifre binarie

Indirizzamento Indiretto con Offset

- Si può applicare la **Tecnica 2** con il modo di indirizzamento indiretto-registro con offset lungo di MIPS32-MARS

Accesso Array 2 in C

```
// arr è allocato
// staticamente
int arr[]={3,-5,1,0,2,6,8,2};
int h=3, k=5;
          ...
arr[h] = arr[k] + 7;
```

Accesso Array 2 in MIPS32

```
# rappresenta int con half
arr: ..half 3,-5,1,0,2,6,8,2
h: .half 3
k: .half 5
          ...
lh        $t0,k # legge k
sll       $t0,$t0,1
lh        $t1,arr($t0) # arr[k]
add       $t1,$t1,7
lh        $t0,h # legge h
sll       $t0,$t0,1
sh        $t1,arr($t0) # arr[h]
```

Indirizzamento Indiretto con Indice

- Nei modi di indirizzamento indiretto con indice, l'indirizzo dell'operando viene calcolato come somma di tre termini
  - Un indirizzo, detto *base*, contenuto in un registro o una parola di memoria
  - Un valore, detto *indice*, contenuto in un registro o una parola di memoria
  - Un valore costante intero, detto *offset*
- La base e l'indice non vengono modificati
- Nella maggior parte dei casi, l'indirizzamento indiretto con indice viene definito nella variante più semplice in cui sia la base che l'indice sono contenuti in registri, che vengono quindi chiamati (nel contesto del modo di indirizzamento), rispettivamente, *registro base* e *registro indice*

Indirizzamento Indiretto con Indice

- MIPS32 non offre modi di indirizzamento di questo tipo
- MC68000 ha un modo di indirizzamento indiretto con indice chiamato *indiretto-registro con indice* in cui
  - La base è il contenuto di uno degli 8 registri indirizzi
  - L'indice è il contenuto, in formato `word` oppure in formato `long`, di uno degli 8 registri indirizzi oppure di uno degli 8 registri dati
  - L'offset è un intero rappresentabile in complemento a 2 con 8 cifre binarie, che viene convertito in intero di 32 cifre binarie mediante **sign-extension** prima di essere sommato alla base e all'indice

Indirizzamento Indiretto con Indice

- 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 Indiretto con Indice

- L'indirizzo dell'operando si ottiene calcolando la somma di tre termini (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 termine che ha come valore
    - 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 vale `w` oppure non è presente

Indirizzamento Indiretto con Indice

- Esempi in MC68000-ASM1

```
        move.l  #8100,a0

        move.w  #500,d0
* l'istruzione seguente copia in d7 la long
* che ha indirizzo di memoria pari a
* 8100 + 500 + 100 = 8700
        move.l  (100,a0,d0.w),d7
* l'esecuzione non modifica a0 e d0

        move.l  #-10,a1
* l'istruzione seguente copia in d7 la word
* che ha indirizzo di memoria pari a
* 8100 - 10 - 30 = 8060
        move.w  (-30,a0,a1.l),d7
* l'esecuzione non modifica a0 e a1
```

Indirizzamento Indiretto con Indice

- Esempi in MC68000-ASM1

```
        move.l  #$00301000,a0

        move.l  #$0020FFF0,d2
* l'istruzione seguente copia in d7 la long
* che ha indirizzo di memoria pari a
* $00301000 + $0020FFF0 + $C = $00510FFC
        move.l  ($C,a0,d2.l),d7
* l'esecuzione non modifica a0 e d2

* l'istruzione seguente copia in d7 la long
* che ha indirizzo di memoria pari a
* $00301000 + $FFFFFFF0 + $C = $00300FFC
        move.l  ($C,a0,d2.w),d7
* l'esecuzione non modifica a0 e d2
```

Indirizzamento Indiretto con Indice

- L'indirizzamento indiretto con indice è utile, in molti *ASM-PM*, per rendere più efficienti operazioni di accesso ad elementi di array in cui sia l'indice sia l'indirizzo dell'area di memoria in cui l'array è memorizzato, possono non essere noti a tempo di traduzione
- Come nel caso dell'indirizzamento indiretto con offset, un singolo uso dell'indirizzamento indiretto con indice permette di effettuare sia l'operazione di somma che conclude il calcolo dell'indirizzo, sia l'operazione di lettura o scrittura in memoria

Indirizzamento Indiretto con Indice

- Come abbiamo visto in precedenza, le possibilità di uso dell'indirizzamento indiretto con offset per tradurre operazioni di accesso ad elementi di array, sono limitate ai casi in cui almeno uno, tra indice e indirizzo dell'area di memoria in cui l'array è memorizzato, siano noti a tempo di traduzione
- L'indirizzamento indiretto con indice permette di superare questa limitazione grazie al fatto che sia la base sia l'indice sono valori contenuti in parole e sono quindi modificabili dinamicamente
- I due addendi che formano l'indirizzo, A e $k \cdot m$ vengono rappresentati uno dalla base e uno dall'indice, di solito
  - A viene rappresentato dalla base
  - $k \cdot m$ viene rappresentato dall'indice

Indirizzamento Indiretto con Indice

- Questa tecnica si può applicare, ad esempio, con il modo di indirizzamento indiretto-registro con indice di MC68000-ASM1

Accesso Array 3 in C

```
// arr è allocato
// staticamente
// o dinamicamente
int h=3, k=5;
          ...
arr[h] = arr[k] + 7;
```

Accesso Array 3 in MC68000

```
* rappresenta int con word
h: dc.w 3
k: dc.w 5
* a0 contiene l'indirizzo di
* un'area di memoria in cui
* è rappresentato l'array
          ...
move.w   k,d1 ; legge k
asl.w    #1,d1
move.w   (a0,d1.w),d2 ; arr[k]
add      #7,d2
move.w   h,a1 ; legge h
add.w    a1,a1
move.w   d2,(a0,a1.w) ; arr[h]
```

Indirizzamento Indiretto con Indice

- Nelle versioni più recenti di M68000, a partire da MC68020, il modo di indirizzamento indiretto con indice 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 termini nel calcolo dell'indirizzo dell'operando, può essere moltiplicato per una piccola costante detta *fattore di scala*

Indirizzamento Indiretto con Indice

- 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 Indiretto con Indice

- L'indirizzo dell'operando si ottiene calcolando la somma di tre termini (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 termine che ha come valore
    - il contenuto della parola di **Ry** di formato `long`, moltiplicato per **s**, se **d** 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 e poi moltiplicato per **s**, se **d** vale `w` oppure non è presente

Indirizzamento Indiretto con Indice

- Esempi in MC68020-ASM1

```
        move.l  #4096,a0

        move.l  #7,d0
* l'istruzione seguente copia in d7 la long
* che ha indirizzo di memoria pari a
* 4096 + 7*2 - 6 = 4104
        move.l  (-6,a0,d0.l*2),d7
* l'esecuzione non modifica a0 e d0

        move.w  #-5,d1
* l'istruzione seguente copia in d7 il byte
* che ha indirizzo di memoria pari a
* 4096 - 5*4 + 3 = 4079
        move.b  (3,a0,d1.w*4),d7
* l'esecuzione non modifica a0 e d1
```

Indirizzamento Indiretto con Indice

- Relativamente alla traduzione di operazioni di accesso ad elementi di array, le nuove caratteristiche dell'indirizzamento indiretto-registro con indice in MC68020, permettono
  - di espandere l'insieme dei casi in cui può essere utile impiegare tale modo di indirizzamento
  - di aumentare ulteriormente i vantaggi, in termini di efficienza, che tale modo di indirizzamento permette di ottenere
- In particolare, la presenza del fattore di scala permette, in alcuni casi, di effettuare mediante l'indirizzamento indiretto-registro con indice
  - l'intero calcolo dell'indirizzo dell'area di memoria che contiene un elemento dell'array
  - l'operazione di lettura o scrittura in memoria

Indirizzamento Indiretto con Indice

- Infatti, per accedere all'elemento di indice k di un array, è necessario calcolare l'indirizzo dell'area di memoria che lo contiene, ovvero $A + k \cdot m$, dove A è l'indirizzo dell'area di memoria che contiene l'intero array e m è la quantità di byte usata per memorizzare ciascuno degli elementi dell'array
- Se m è pari a uno dei valori ammissibili come fattore di scala, si può calcolare $A + k \cdot m$ attraverso l'indirizzamento indiretto-registro con indice rappresentando
  - A con la base
  - k con l'indice

Indirizzamento Indiretto con Indice

- Nell'esempio precedente, l'uso del fattore di scala permette, nel codice MC68020-ASM1, di evitare due istruzioni rispetto alla controparte MC68000-ASM1

Accesso Array 3 in C

```
// arr è allocato
// staticamente
// o dinamicamente
int h=3, k=5;
          ...
arr[h] = arr[k] + 7;
```

Accesso Array 3 in MC68020

```
* rappresenta int con word
h: dc.w 3
k: dc.w 5
* a0 contiene l'indirizzo in
* cui è rappresentato l'array
          ...
move.w   k,d1 ; legge k
* rimossa asl.w #1,d1
move.w   (a0,d1.w*2),d2
add      #7,d2
move.w   h,a1 ; legge h
* rimossa add.w a1,a1
move.w   d2,(a0,a1.w*2)
```