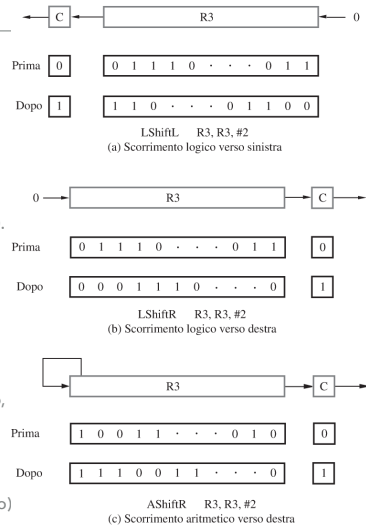
Scorrimento

- Operazioni per far scorrere (shift) a destra o a sinistra i bit di un operando
- `LShiftL` (logic shift left) è l'istruzione per lo scorrimento logico a sinistra di un operando per un certo numero di bit

```
LShiftL Ri, Rj, contatore
```

- fa scorrere il contenuto di Rj per un certo numero contatore e pone il risultato in Ri. Le posizioni che si liberano a destra sono riempite con 0. I bit a sinistra che scorrono sono posti uno a uno nel bit di riporto (carry) e scartati via via, tranne l'ultimo a uscire
- Analogamente per `LShiftR` (logic shift right)
- Lo scorrimento può servire a impaccare cifre decimali, ovvero a usare un unico byte per memorizzare due cifre decimali di 4 bit ciascuna
- Far scorrere a sinistra di una posizione significa raddoppiare il numero, far scorrere a destra significa dimezzare
- Se i bit rappresentano numeri relativi con rappresentazione in complemento a due, quando si scorre il bit più significativo deve essere replicato, a questo scopo si usa `AShiftR` (scorrimento aritmetico)



Rotazione

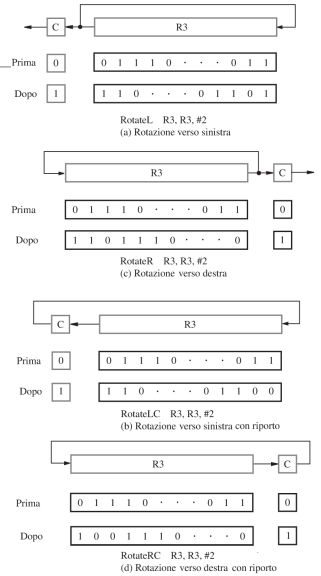
- Analogamente allo scorrimento, si possono far rientrare i bit che scorrono da un estremo all'estremo opposto, così da non perdere niente
- La rotazione può avvenire girando verso sinistra

```
RotateL R3, R3, #2
```

- Girando verso destra

```
RotateR R3, R3, #2
```

- La rotazione copierà il bit uscente sul riporto, prima di farlo rientrare dall'altro estremo
- Oppure, la rotazione può coinvolgere il bit di riporto che si aggiunge come posizione ulteriore di rotazione `RotateLC R3, R3, #2`



Moltiplicazione e divisione

- Per moltiplicare due fattori con segno, quindi rappresentati in complemento a due

```
Multiply Rk, Ri, Rj
```

- L'istruzione sopra effettua l'operazione `Rk ← [Ri] * [Rj]`
- In generale occorrono 2n bit per rappresentare il prodotto di due numeri a n bit. Es. 15*15 = 225, ovvero 1111 x 1111 = 1110 0001
- Può succedere che il prodotto non stia tutto nel registro Rk. Nella maggior parte dei linguaggi macchina la moltiplicazione scrive solo gli n bit meno significativi del prodotto
- Oppure, sono previste varianti per scrivere il risultato in una coppia di registri, in Rk+1 va la metà più significativa
- Per la divisione

```
Divide Rk, Ri, Rj
```

- L'istruzione sopra effettua l'operazione `Rk ← [Ri] / [Rj]`
- Il resto della divisione viene scritto in Rk+1, o non viene calcolato

Valori immediati a 32 bit (S. 2.9)

- I modi di indirizzamento immediato e assoluto di un processore in stile RISC limitano a 16 bit lo spazio disponibile per l'operando o il suo indirizzo, rispettivamente, nel formato dell'istruzione che occupa complessivamente 32 bit
- Se i 16 bit non bastano a codificare l'operando o il suo indirizzo il processore RISC si serve di istruzioni che effettuano OR logici, per estendere l'operando immediato a 16 bit e mettendo a zero i bit di ordine più alto, ottenendo 32 bit

```
Or Rdst, Rsrc, #Valore
```

- E inoltre, per formare 32 bit mettendo i 16 bit dell'operando immediato nei bit di ordine più alto, e zero nei bit di ordine più basso

```
OrHigh Rdst, Rsrc, #Valore
```

- Quindi, il valore di 32 bit 0x20004FF0 si può caricare con la seguente sequenza

```
OrHigh R2, R0, #0x2000
Or R2, R2, #0x4FF0
```

- Per facilitare la stesura di programmi, sono disponibili alcune pseudoistruzioni che sono poi sostituite dall'assemblatore in corrispondenti sequenze di istruzioni macchina

Insiemi di istruzioni CISC (S. 2.10)

- Gli insiemi di istruzioni CISC non sono vincolati all'architettura load/store (carica/immagazzina), nella quale si possono effettuare operazioni aritmetiche e logiche solo su operandi che si trovano nei registri del processore
- Inoltre, le istruzioni non necessariamente devono occupare una singola parola di memoria, alcune istruzioni possono occupare più parole di memoria
- Le istruzioni usano il formato a due indirizzi

```
Add B, A
```

- effettua l'operazione `B ← [A] + [B]` su operandi in memoria e il risultato è inviato in memoria. La locazione di memoria B è sia una sorgente che una destinazione
- Un'istruzione C = A + B può essere effettuata con la sequenza

```
Move C, B
Add C, A
```

- Il formato dell'istruzione Move è `Move destinazione, sorgente`, dove destinazione e sorgente possono essere sia registri che locazioni di memoria. In alcuni processori CISC un operando deve essere in un registro, mentre l'altro può essere in memoria, allora per l'istruzione C = A + B si avrebbe la sequenza

```
Move Ri, A
Add Ri, B
Move C, Ri
```


Indirizzamenti ulteriori

- Oltre ai modi di indirizzamento visti per i processori RISC, i processori CISC aggiungono altri modi di indirizzamento
  - Con **autoincremento** e **autodecremento**, l'indirizzo dell'operando è contenuto in un registro di processore e dopo aver calcolato l'indirizzo effettivo il contenuto del registro viene modificato. Questo facilita l'accesso al successivo elemento
  - Per l'autoincremento si usa `(Ri)+` e per l'autodecremento si usa `–(Ri)`
- Quanto viene aggiunto o decrementato è stabilito dal processore in base alla dimensione del dato, e l'istruzione codifica questa informazione
- Per impilare un nuovo elemento si può usare

```
Move –(SP), ELEMENTO
```

- Per togliere dalla pila si usa

```
Move ELEMENTO, (SP)+
```


Indirizzamento con modo relativo a PC

- Il modo con indice e spiazzamento può essere applicato al PC ottenendo il **modo relativo** al contatore di programma (PC)
- X(PC) indica una parola di memoria traslata di X byte rispetto all'indirizzo contenuto in PC, quindi `EA = X + [PC]`
- L'indirizzo è calcolato come nel modo con indice e spiazzamento, prendendo il registro PC come indice

Bit di esito o condizione

- Il processore tiene traccia di alcune informazioni relative all'esito di varie operazioni. Tali informazioni sono espresse in un insieme di bit chiamati *codici o bit di condizione* (*condition code flag*), o *bit di esito*
- I bit di esito sono raggruppati in un registro interno del processore chiamato *registro di stato* (*status register*)
- I bit di esito più comuni sono
  - N (negativo) vale 1 se il risultato è negativo, 0 altrimenti
  - Z (zero) vale 1 se il risultato è nullo, 0 altrimenti
  - V (trabocco) vale 1 se vi è trabocco in complemento a due, 0 altrimenti (da oVerflow)
  - C (riporto) vale 1 se trabocco in binario naturale, 0 altrimenti (da Carry)
- Tutti i bit vengono aggiornati quando si esegue un'istruzione aritmetica o logica. I bit N e Z vengono aggiornati anche quando si esegue un trasferimento di un dato
- L'istruzione Branch>0 CICLO non ha bisogno di specificare il registro, salta se né N né Z sono 1

Stili RISC e CISC (S. 2.11)

- Caratteristiche dello stile RISC
  - modi di indirizzamento semplici
  - tutte le istruzioni occupano una sola parola
  - meno istruzioni
  - operazioni aritmetiche e logiche possono essere effettuate solo sui registri del processore
  - istruzioni semplici favoriscono l'esecuzione veloce
  - programmi tendenzialmente di dimensioni più grandi
- Caratteristiche dello stile CISC
  - modi di indirizzamento più complessi
  - istruzioni più complesse che possono occupare più parole di memoria
  - tante istruzioni che effettuano compiti complessi
  - operazioni aritmetiche e logiche possono usare operandi in memoria e nei registri del processore
  - trasferimenti da una locazione di memoria a un'altra tramite una singola istruzione
  - programmi tendenzialmente di dimensioni più piccole

Codifica di istruzioni (S. 2.13)

- Il linguaggio simbolico per essere eseguibile da parte del processore va rappresentato o codificato (encoded) sotto forma numerica binaria (è il compito svolto dall'assemblatore)
- Per es. l'istruzione `Add Rdst, Rsrc1, Rsrc2` è rappresentativa di una categoria che ha tre indirizzi per gli operandi
- Se il processore ha 32 registri occorre usare 5 bit per specificare ciascun registro. Con un'istruzione codificata in 32 bit, si hanno 32-15=17 bit per codificare il codice operativo che indica l'operazione
- Per `Add Rdst, Rsrc1, #Valore` Si hanno 10 bit per specificare i due registri, dei rimanenti 22 bit, 16 bit si usano per l'operando immediato, e 6 bit per il codice operativo.
- Le istruzioni `Load` e `Store` possono usare questo formato dove il modo di indirizzamento con indice e spiazzamento usa i 16 bit del valore immediato per lo spiazzamento
- Le istruzioni di salto condizionale possono usare questo formato, i bit del valore immediato saranno usati per la destinazione di salto, di solito come differenza rispetto alla corrente posizione del `PC`
- Il terzo formato è usato per le istruzioni di salto a sottoprogramma

(a) Formato con operandi in registri

(b) Formato con operando immediato

(c) Formato per chiamata