7 Appendice: istruzioni Assembler aggiuntive

7.1.1 ADD WITH CARRY

FORMATO: `ADC source, destination`

AZIONE: Modifica l'operando destinatario sommandovi sia l'operando sorgente sia il contenuto del flag CF (il risultato dell'operazione è consistente sia se gli operandi sono interpretati come numeri naturali sia se gli operandi sono interpretati come numeri interi); mette ad 1 il contenuto del flag CF se, interpretando gli operandi come numeri naturali, si è verificato un riporto; mette ad 1 il contenuto del flag OF se, interpretando gli operandi come numeri interi, si è verificato un traboccamento. Poiché l'istruzione ADC prevede la gestione del riporto che può essere stato generato durante l'esecuzione di una precedente istruzione ADD o ADC, essa può essere usata per predisporre pacchetti di istruzioni per la somma di operandi multi_word.

Ad esempio, la somma di due operandi a 64 bit (16 cifre esadecimali) può essere effettuata utilizzando una istruzione ADD per elaborare i 32 bit meno significativi ed una istruzione ADC per elaborare i 32 bit più significativi.

```
   ADC      ADD
56A9C2D4 67A43B5F +
44B9A5A4 A6B4C55A =
9B636879 0E5900B9
```

FLAG di cui viene modificato il contenuto: Tutti.

Operandi	Esempi
Memoria, Registro Generale	ADC 0x00002000,%EDX
Registro Generale, Memoria	ADC %CL,0x12AB1024
Registro Generale, Registro Generale	ADC %AX,%DX
Immediato, Memoria	ADCB $0x5B,(%EDI)
Immediato, Registro Generale	ADC $0x54A3,%AX

7.1.2 SUBTRACT WITH BORROW

FORMATO: `SBB source, destination`

AZIONE: Modifica l'operando destinatario sottraendovi sia l'operando sorgente sia il contenuto del flag CF (il risultato dell'operazione è consistente sia se gli operandi sono interpretati come numeri naturali sia se gli operandi sono interpretati come numeri interi); mette ad 1 il contenuto del flag CF se, interpretando gli operandi come numeri naturali, è stato richiesto un prestito; mette ad 1 il contenuto del flag OF se, interpretando gli operandi come numeri interi, si è verificato un traboccamento. Poiché l'istruzione SBB prevede la gestione del prestito che può essere stato richiesto durante l'esecuzione di una precedente istruzione SUB o SBB, essa può essere usata per predisporre pacchetti di istruzioni per la sottrazione di operandi multi_word.

Ad esempio, la somma di due operandi a 64 bit (16 cifre esadecimali) può essere effettuata utilizzando una SUB per elaborare i 32 bit meno significativi ed una SBB per elaborare i 32 bit più significativi.

```
  SBB       SUB
9B636879 0E5900B9 -
56A9C2D4 67A43B5F =
44B9A5A4 A6B4C55A
```

FLAG di cui viene modificato il contenuto: Tutti.

Operandi	Esempi
Memoria, Registro Generale	SBB 0x00002000,%EDX
Registro Generale, Memoria	SBB %CL,0x12AB1024
Registro Generale, Registro Generale	SBB %AX,%DX
Immediato, Memoria	SBBW $0x255B,(%EDI)
Immediato, Registro Generale	SBB $0x54A3,%AX

7.1.3 ROTATE THROUGH CARRY LEFT/RIGHT

FORMATO: `RCL source, destination`

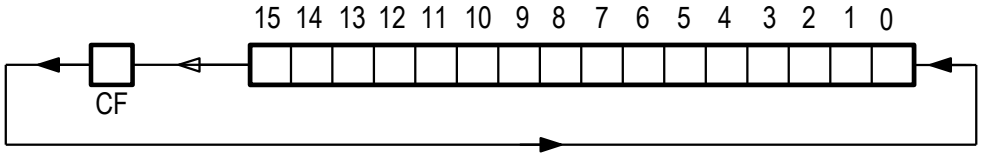
`RCL destination`

`RCR source, destination`

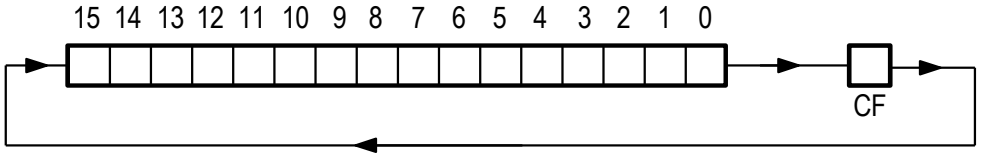
`RCR destination`

Fanno la stessa cosa delle corrispondenti istruzioni di rotazione, coinvolgendo anche CF. I formati di indirizzamento sono identici.

LEFT:



RIGHT:



Operandi	Esempi
Immediato, Registro Generale	ROR $1,%EAX
Immediato, Memoria	RORB $7,0x00002000
Registro CL, Registro Generale	ROR %CL,%BX
Registro CL, Memoria	RORL %CL,(%EDI)
Memoria	RORL (%EDI)
Registro Generale	ROR %AX

7.1.4 LOOP / LOOPcon

FORMATO: `LOOP %EIP ± displacement`

`LOOPcon %EIP ± displacement`

AZIONE: Decrementa `ECX`. Se, dopo il decremento, `ECX≠ 0,` e se l'eventuale condizione *con* è vera, sostituisce `EIP` con il nuovo indirizzo.

FLAG di cui viene modificato il contenuto: Nessuno.

Di seguito sono riassunti i codici operativi delle istruzioni di `LOOP` condizionato e, per ciascuno di essi, è spiegato brevemente il significato della pertinente condizione *con*.

LOOPE LOOPZ	(Loop if Equal, Loop if Zero) la condizione è soddisfatta se ZF contiene 1; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era uguale all'operando sorgente.
LOOPNE LOOPNZ	(Loop if Not Equal, Loop if Not Zero) la condizione è soddisfatta se ZF contiene 0; il verificarsi di questa condizione dopo un'istruzione CMP indica che l'operando destinatario non era uguale all'operando sorgente.

7.1.5 MOVE DATA FROM STRING TO STRING (with REPEAT)

FORMATO: `MOVSsuf`
`REP MOVSsuf`

AZIONE: Copia il numero di byte specificato dal suffisso *suf* dall'indirizzo di memoria puntato da `ESI` all'indirizzo di memoria puntato da `EDI`. Se `DF=0`, somma ad `ESI` e ad `EDI` il numero di byte specificato dal suffisso. Se `DF=1`, sottrae da `ESI` e da `EDI` il numero di byte specificato dal suffisso.
Se viene premesso il prefisso `REP`, allora le azioni indicate sopra vengono ripetute per il numero di volte specificato in `ECX`, che viene decrementato fino a zero.
FLAG di cui viene modificato il contenuto: Nessuno.

7.1.6 LOAD STRING

FORMATO: `LODSsuf`

AZIONE: Copia il numero di byte specificato dal suffisso *suf* dall'indirizzio di memoria puntato da `ESI` dentro `AL`, `AX` o `EAX`. Se `DF=0`, somma a `ESI` il numero di byte specificato dal suffisso. Se `DF=1`, sottrae da `ESI` il numero di byte specificato dal suffisso.
Se viene premesso il prefisso `REP`, allora le azioni indicate sopra vengono ripetute per il numero di volte specificato in `ECX`, che viene decrementato fino a zero.
FLAG di cui viene modificato il contenuto: Nessuno.

7.1.7 STORE STRING (with REPEAT)

FORMATO: `STOSsuf`
`REP STOSsuf`

AZIONE: Copia il contenuto di `AL`, `AX` o `EAX` (a seconda del suffisso *suf*) all'indirizzo di memoria puntato da `EDI`. Se `DF=0`, somma a `EDI` il numero di byte specificato dal suffisso. Se `DF=1`, sottrae da `EDI` il numero di byte specificato dal suffisso.
Se viene premesso il prefisso `REP`, allora le azioni indicate sopra vengono ripetute per il numero di volte specificato in `ECX`, che viene decrementato fino a zero.
FLAG di cui viene modificato il contenuto: Nessuno.

7.1.8 COMPARE STRING OPERANDS (with CONDITIONAL REPEAT)

FORMATO: `CMPSsuf`
`REPE CMPSsuf`
`REPNE CMPSsuf`

AZIONE: Confronta due operandi in memoria, lunghi il numero di byte specificato dal suffisso *suf*. L'operando sorgente si trova a partire dall'indirizzo di memoria puntato da `ESI`, l'operando destinatario si trova a partire dall'indirizzo di memoria puntato da `EDI`. Se `DF=0`, somma a `ESI` e a `EDI` il numero di byte specificato dal suffisso. Se `DF=1`, sottrae da `ESI` e da `EDI` il numero di byte specificato dal suffisso.
Se tale istruzione viene seguita da un'istruzione di salto condizionato `Jcon`, la condizione *con* si intende riferita all'operando *sorgente* (quello puntato da `ESI`), contrariamente a quanto avviene con l'istruzione `CMP`.
Se viene premesso il prefisso `REPE (REPNE)`, allora le azioni indicate sopra vengono ripetute per il numero massimo di volte specificato in `ECX`, che viene decrementato fino a zero. Le ripetizioni proseguono finché gli operandi sono uguali (diversi).
FLAG di cui viene modificato il contenuto: Tutti.

7.1.9 SCAN STRING (with CONDITIONAL REPEAT)

FORMATO: `SCASsuf`
`REPE SCASsuf`
`REPNE SCASsuf`

AZIONE: Confronta il contenuto di `AL`, `AX` o `EAX` (a seconda del suffisso *suf*) con l'operando destinatario di pari lunghezza che si trova in memoria a partire dall'indirizzo puntato da `EDI`. Se `DF=0`, somma a `EDI` il numero di byte specificato dal suffisso. Se `DF=1`, sottrae da `EDI` il numero di byte specificato dal suffisso.
Se tale istruzione viene seguita da un'istruzione di salto condizionato `Jcon`, la condizione *con* si intende riferita all'operando *destinatario* (quello puntato da `EDI`).
Se viene premesso il prefisso `REPE (REPNE)`, allora le azioni indicate sopra vengono ripetute per il numero massimo di volte specificato in `ECX`, che viene decrementato fino a zero. Le ripetizioni proseguono finché gli operandi sono uguali (diversi).
FLAG di cui viene modificato il contenuto: Tutti, secondo l'algoritmo della `CMP`, ad ogni ripetizione.

7.1.10 INPUT STRING (with REPEAT)

FORMATO: `INSsuf`
`REP INSsuf`

AZIONE: preleva uno, due, quattro byte (a seconda del suffisso *suf*) dalla porta, doppia porta, quadrupla porta di ingresso il cui offset è contenuto in `DX`. Il dato prelevato viene inserito in memoria a partire dall'indirizzo di memoria contenuto in `EDI`. Se `DF=0`, somma a `EDI` il numero di byte specificato dal suffisso. Se `DF=1`, sottrae da `EDI` il numero di byte specificato dal suffisso.
Se viene premesso il prefisso `REP`, allora le azioni indicate sopra vengono ripetute per il numero di volte specificato in `ECX`, che viene decrementato fino a zero.
FLAG di cui viene modificato il contenuto: Nessuno

7.1.11 OUTPUT STRING (with REPEAT)

FORMATO: `OUTSsuf`
`REP OUTSsuf`

AZIONE: fa uscita di uno, due, quattro byte (a seconda del suffisso *suf*) alla porta, doppia porta, quadrupla porta di uscita il cui offset è contenuto in `DX`. L'operando sorgente è prelevato a partire dall'indirizzo di memoria contenuto in `ESI`. Se `DF=0`, somma a `ESI` il numero di byte specificato dal suffisso. Se `DF=1`, sottrae da `ESI` il numero di byte specificato dal suffisso.
Se viene premesso il prefisso `REP`, allora le azioni indicate sopra vengono ripetute per il numero di volte specificato in `ECX`, che viene decrementato fino a zero.
FLAG di cui viene modificato il contenuto: Nessuno

7.1.12 SET /CLEAR DIRECTION FLAG

FORMATO: `STD`
`CLD`
AZIONE: setta (`STD`) o resetta (`CLD`) il flag `DF`
FLAG di cui viene modificato il contenuto: `DF`.

Quando `DF` vale 1, le istruzioni stringa decrementano i registri indice (`ESI` e/o `EDI`).
Quando `DF` vale 0, le istruzioni stringa incrementano i registri indice (`ESI` e/o `EDI`).

7.1.13 SET if CONDITION MET

FORMATO: `SETcon %reg`

`SETcon mem`

AZIONE: Setta a `1` o a `0` l'operando destinatario, a seconda se la condizione `con` specificata è vera o falsa. Le condizioni sono le stesse che possono essere scritte come salti condizionati. Il destinatario deve essere ad 8 bit.

FLAG di cui viene modificato il contenuto: Nessuno.

Di seguito sono riassunti i codici operativi di alcune delle istruzioni di salto condizionato e, per ciascuno di essi, è spiegato brevemente il significato della pertinente condizione *con*.

SETE	(Set if Equal) la condizione è soddisfatta se ZF contiene 1; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era uguale all'operando sorgente.
SETNE	(Set if Not Equal) la condizione è soddisfatta se ZF contiene 0; il verificarsi di questa condizione dopo un'istruzione CMP indica che l'operando destinatario non era uguale all'operando sorgente.
SETA	(Set if Above) la condizione è soddisfatta se CF contiene 0 e ZF contiene 0; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era maggiore dell'operando sorgente, essendo gli operandi interpretati come numeri naturali.
SETAE	(Set if Above or Equal) la condizione è soddisfatta se CF contiene 0; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era maggiore o uguale rispetto all'operando sorgente, essendo gli operandi interpretati come numeri naturali.
SETB	(Set if Below) la condizione è soddisfatta se CF contiene 1; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era minore dell'operando sorgente, essendo gli operandi interpretati come numeri naturali.
SETBE	(Set if Below or Equal) la condizione è soddisfatta se CF contiene 1 o ZF contiene 1; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era minore od uguale rispetto all'operando sorgente, essendo gli operandi interpretati come numeri naturali.
SETG	(Set if Greater) la condizione è soddisfatta se ZF contiene 0 e se il contenuto di SF è uguale a quello di OF; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era maggiore dell'operando sorgente, essendo gli operandi interpretati come numeri interi.
SETGE	(Set if Greater or Equal) la condizione è soddisfatta se il contenuto di SF è uguale a quello di OF; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario maggiore o uguale rispetto all'operando sorgente, essendo gli operandi interpretati come numeri interi.
SETL	(Set if Less) la condizione è soddisfatta se il contenuto di SF è diverso da quello di OF; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era minore dell'operando sorgente, essendo gli operandi interpretati come numeri interi.
SETLE	(Set if Less or Equal) la condizione è soddisfatta se ZF contiene 1 oppure se il contenuto di SF è diverso da quello di OF; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era minore od uguale rispetto all'operando sorgente, essendo gli operandi interpretati come numeri interi.
SETZ	(Set if Zero) la condizione è soddisfatta se ZF contiene 1; il verificarsi di questa condizione indica che il risultato dell'istruzione precedente è stato zero.
SETNZ	(Set if Not Zero) la condizione è soddisfatta se ZF contiene 0; il verificarsi di questa condizione indica che il risultato dell'istruzione precedente è stato diverso da zero.
SETC	(Set if Carry) la condizione è soddisfatta se CF contiene 1.
SETNC	(Set if No Carry) la condizione è soddisfatta se CF contiene 0.
SETO	(Set if Overflow) la condizione è soddisfatta se OF contiene 1.
SETNO	(Set if No Overflow) la condizione è soddisfatta se OF contiene 0.
SETS	(Set if Sign) la condizione è soddisfatta se SF contiene 1.
SETNS	(Set if No Sign) la condizione è soddisfatta se SF contiene 0.

7.1.14 CONDITIONAL MOVE

FORMATO: `CMOVcon %reg, %reg`
`CMOVcon mem, %reg`

AZIONE: esegue la corrispondente istruzione `MOV` se è vera la condizione specificata, altrimenti non fa niente. Sorgente e destinatario devono essere a 16 o 32 bit.

FLAG di cui viene modificato il contenuto: Nessuno.

Di seguito sono riassunti i codici operativi **di alcune** delle istruzioni di conditional move e, per ciascuno di essi, è spiegato brevemente il significato della pertinente condizione *con*.

CMOVE	(Move if Equal) la condizione è soddisfatta se ZF contiene 1; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era uguale all'operando sorgente.
CMOVNE	(Move if Not Equal) la condizione è soddisfatta se ZF contiene 0; il verificarsi di questa condizione dopo un'istruzione CMP indica che l'operando destinatario non era uguale all'operando sorgente.
CMOVA	(Move if Above) la condizione è soddisfatta se CF contiene 0 e ZF contiene 0; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era maggiore dell'operando sorgente, essendo gli operandi interpretati come numeri naturali.
CMOVAE	(Move if Above or Equal) la condizione è soddisfatta se CF contiene 0; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era maggiore o uguale rispetto all'operando sorgente, essendo gli operandi interpretati come numeri naturali.
CMOVB	(Move if Below) la condizione è soddisfatta se CF contiene 1; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era minore dell'operando sorgente, essendo gli operandi interpretati come numeri naturali.
CMOVBE	(Move if Below or Equal) la condizione è soddisfatta se CF contiene 1 o ZF contiene 1; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era minore od uguale rispetto all'operando sorgente, essendo gli operandi interpretati come numeri naturali.
CMOVG	(Move if Greater) la condizione è soddisfatta se ZF contiene 0 e se il contenuto di SF è uguale a quello di OF; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era maggiore dell'operando sorgente, essendo gli operandi interpretati come numeri interi.
CMOVGE	(Move if Greater or Equal) la condizione è soddisfatta se il contenuto di SF è uguale a quello di OF; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario maggiore o uguale rispetto all'operando sorgente, essendo gli operandi interpretati come numeri interi.
CMOVL	(Move if Less) la condizione è soddisfatta se il contenuto di SF è diverso da quello di OF; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era minore dell'operando sorgente, essendo gli operandi interpretati come numeri interi.
CMOVLE	(Move if Less or Equal) la condizione è soddisfatta se ZF contiene 1 oppure se il contenuto di SF è diverso da quello di OF; il verificarsi di questa condizione dopo l'esecuzione di un'istruzione CMP indica che l'operando destinatario era minore od uguale rispetto all'operando sorgente, essendo gli operandi interpretati come numeri interi.
CMOVZ	(Move if Zero) la condizione è soddisfatta se ZF contiene 1; il verificarsi di questa condizione indica che il risultato dell'istruzione precedente è stato zero.
CMOVNZ	(Move if Not Zero) la condizione è soddisfatta se ZF contiene 0; il verificarsi di questa condizione indica che il risultato dell'istruzione precedente è stato diverso da zero.
CMOVC	(Move if Carry) la condizione è soddisfatta se CF contiene 1.
CMOVNC	(Move if No Carry) la condizione è soddisfatta se CF contiene 0.
CMOVO	(Move if Overflow) la condizione è soddisfatta se OF contiene 1.
CMOVNO	(Move if No Overflow) la condizione è soddisfatta se OF contiene 0.
CMOVS	(Move if Sign) la condizione è soddisfatta se SF contiene 1.
CMOVNS	(Move if No Sign) la condizione è soddisfatta se SF contiene 0.