Manuale utente della FPU di EduMIPS64

Versione 1, 23 Settembre 2007

di Massimo Trubia
massimotrubia83@gmail.com

Capitolo 1

1.1 Introduzione

Questo manuale utente completa quello ufficiale di EduMIPS64 con il capitolo relativo all'utilizzo della floating point unit. Nel Paragrafo 1.2 viene introdotto il formato double, i valori speciali floating point definiti dallo standard che ne disciplina il trattamento nei calcolatori (IEEE 754), e le condizioni di eccezioni che i calcoli floating point possono provocare.

Nel Paragrafo 1.3 viene illustrato come EduMIPS64 consente di attivare e disattivare le trap relative alle condizioni di eccezione IEEE.

Nel Paragrafo 1.4 si parla del modo in cui i valori double e i valori speciali vengono accettati da EduMIPS64 per essere caricati in memoria.

Nel Paragrafo 1.5 è introdotto il registro FCSR, usato dalla FPU per autogestirsi. In esso vengono memorizzate le informazioni riguardanti l'arrotondamento, i risultati booleani delle istruzioni di confronto e le politiche di gestione delle eccezioni IEEE.

Infine, nel Paragrafo 1.6, sono elencate tutte le istruzioni dell'ISA MIPS64 implementate in EduMIPS64.

1.2 Valori speciali

L'aritmetica in virgola mobile dei calcolatori è caratterizzata dal fatto che, anche in presenza di un'operazione matematica non valida, si potrebbe scegliere di continuare la computazione ignorando quanto è accaduto. In questo scenario, operazioni come divisioni tra zeri, oppure radici quadrate di numeri negativi devono generare comunque un risultato che, non essendo un numero (Not a Number), è trattato come qualcosa di diverso. Prima di continuare questa trattazione, vengono definiti i numeri in virgola mobile (floating point) a doppia precisione. Vengono chiamati

double e possono rappresentare valori compresi nel dominio $[-1.79E308, -4.94E-324] \cup \{0\} \cup [4.94E-324, 1.79E308]$.

1.2.1 NaN o Invalid Operation

Lo standard IEEE 754, il quale regolamenta la manipolazione dei numeri floating point nei calcolatori, ha definito che le operazioni matematiche non valide possono sia provocare una segnalazione durante l'esecuzione del programma (trap per la condizione di eccezione IEEE **Invalid Operation**), che fornire, come risultato, il valore speciale QNaN (Quit Not a Number). Un altro valore NaN, che genera incondizionatamente la stessa trap appena viene rilevato come operando, è SNaN (Signalling NaN). Tale valore è raramente utilizzato nelle applicazioni, e storicamente è stato usato per inizializzare le variabili.

1.2.2 Zeri o Underflow

Un altro valore speciale definito dallo standard è lo zero. Dal momento che il formato double non include lo zero nel dominio dei valori rappresentati, esso è considerato alla stregua di un valore speciale. Esistono uno zero positivo e uno negativo: quando si tenta di rappresentare un valore negativo molto vicino allo zero, ma fuori dal dominio rappresentabile dai double ($\in]-4.94E-324, 0[$),
e si desidera a tutti i costi un risultato (non una trap per la condizione di eccezione **Underflow**), allora il numero restituito è -0. Analogamente, viene restituito $+0$ se si tenta di rappresentare un numero nell'intervallo $[0, 4.94E-324[$, e non si è preferito avere la stessa trap.

1.2.3 Infiniti od Overflow

Quando si tenta di rappresentare un numero estremamente grande ($\in]1.79E308, +\infty[$), o estrememente piccolo ($\in]-\infty, -1.79E308[$), al di fuori cioè del dominio rappresentabile dal formato double, vengono restituiti i valori speciali $+\infty$, nel primo caso, e $-\infty$ nel secondo caso. In alternativa, si può avere una trap per via della condizione di eccezione **Overflow**.

1.2.4 Infiniti o Divide by zero

Gli infiniti potrebbero anche essere restituiti da una divisione per zero, nelle quali si effettua il prodotto tra il segno dello zero e quello del divisore, per restituire un opportuno infinito. Nel

caso in cui non si voglia alcun valore restituito, si verifica la trap per la condizione di eccezione **Divide by zero**.

1.3 Configurazione delle eccezioni

EduMIPS64 consente di abilitare o disabilitare le trap relative alle 4 delle 5 condizioni di eccezione IEEE, implementate dalla scheda *Eccezioni FPU* della finestra *Configura→ Impostazioni*. Se esse sono disabilitate, verrà fornito un risultato per qualunque operazione speciale la FPU effettui (si veda il Paragrafo 1.2). Nel caso illustrato in Figura 1.1, in cui alcune caselle di controllo sono spuntate, se la CPU non maschera le eccezioni sincrone nel simulatore (Figura 1.2), verrà simulata una trap relativa alla condizione di eccezione IEEE che si è verificata (Figura 1.3).

Figura 1.1: Configurazione delle trap per le eccezioni IEEE

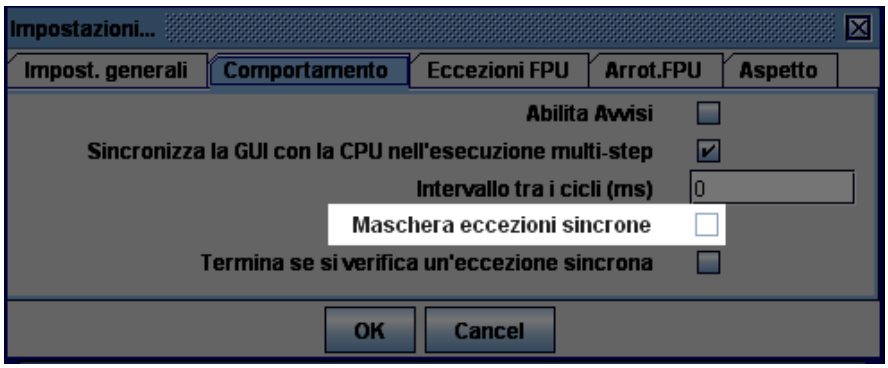

Figura 1.2: Opzione che maschera le eccezioni sincrone(disabilita tutte le trap)

Figura 1.3: Finestra che notifica la trap

1.4 Direttiva .double

La direttiva `.double`, da inserire nella sezione `.data` del file sorgente (.s), consente di allocare una cella della memoria di EduMIPS64, dove inserire un valore formattato *double*. Le sintassi valide del comando sono

```
nome_variabile: .double numero_double
nome_variabile: .double parola_chiave
```

dove `numero_double` può essere rappresentato sia in forma estesa (`1.0,0.003`), oppure in notazione scientifica (`3.7E-12,0.5E32`). Invece, `parola_chiave` può assumere i valori `POSITIVEINFINITY, NEGATIVEINFINITY,`
`POSITIVEZERO, NEGATIVEZERO, SNAN` e `QNAN`, consentendo l'inserimento diretto in memoria dei valori speciali.

1.5 Registro FCSR

L'FCSR(Floating point Control Status Register) è il registro che controlla i diversi aspetti funzionali della FPU. Esso è lungo 32 bit e, fino alla ridisegnazione grafica di EduMIPS64, sarà posto nella finestra delle statistiche. Il campo **FCC** è costituito da 8 bit, identificati con numeri da 0 a 7. Le istruzioni condizionali (`C.EQ.D,C.LT.D`) lo utilizzano per memorizzare il risultato booleano di un confronto tra due registri. I campi Cause, Enables e Flag gestiscono la dinamica delle eccezioni IEEE, illustrate nel Paragrafo 1.2. Essi sono costituiti, ognuno, da 5 bit identificati con le lettere V (Invalid operation), Z (Divide by zero), O (Overflow),U (Underflow) e I (Inexact); quest'ultimo bit non viene al momento utilizzato.

Il campo **Cause** indica se si è verificata una qualunque eccezione IEEE durante la simu-

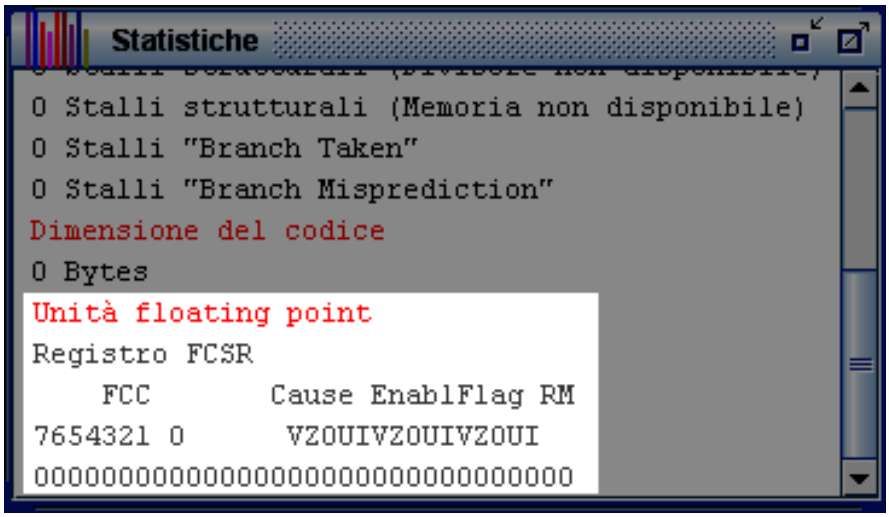

Figura 1.4: Registro FCSR in EduMIPS64

lazione, presentando un 1 nel relativo bit. È utile quando si esegue un programma dall'inizio alla fine senza fermarlo, per determinare se si è verificata una qualunque eccezione.

Il campo **Enable** mostra le eccezioni IEEE per le quali è attiva la trap. I bit di questo campo vengono modificati, anche senza resettare il simulatore, dalla finestra di configurazione della Figura 1.1.

Il campo **Flag** mostra le eccezioni IEEE che si sono verificate ma, non avendo la relativa trap attivata, hanno fornito come risultato dei valori speciali, illustrati nel Paragrafo 1.2.

Il campo **RM** mostra la modalità di arrotondamento corrente usata, in EduMIPS64, per le istruzioni che convertono numeri floating point in interi (si veda l'istruzione CVT.L.D per ulteriori dettagli).

1.6 Instruction set

Per una consultazione efficiente, le istruzioni dell'ISA MIPS64, implementate in EduMIPS64, vengono elencate in ordine alfabetico. Le operazioni eseguite vengono rappresentate mediante uno pseudocodice in cui l'$i-$esima cella di memoria è indicata con `memory[i]`, i bit del campo FCC del registro FCSR mediante `FCSR_FCC[cc]`, con $cc \in [0, 7]$. In alcune istruzioni, per evitare ambiguità, i registri sono indicati come `GPR[i]` e `FPR[i]`, con $i \in [0, 31]$, ma nella maggior parte dei casi essi vengono indicati qualitativamente con la notazione `rx` o `fx`,

dove $x \in \{d, s, t\}$. Le tre lettere servono solo a distinguere, al più, tre registri per ogni istruzione. Infine, i valori ritornati dalle operazioni di conversione vengono indicati con la notazione `convert_tipoconversione(registro[,tipo arrotondamento])`, dove il parametro tra parentesi quadre è presente solo in certe circostanze.

Per prendere confidenza con le istruzioni floating point, alcuni file sorgenti possono essere scaricati dal link

`http://www.edumips.org/attachment/wiki/Upload/FPUMaxSamples.rar`.

ADD.D

Sintassi: `ADD.D fd, fs, ft`
Descrizione: `fd = fs + ft`
Eccezioni: Le trap di Overflow e Underflow vengono generate se il risultato non può essere rappresentato secondo lo standard IEEE 754. Invalid Operation è generata se fs o ft contengono QNaN o SNaN, o se viene eseguita un'operazione non valida ($+\infty - \infty$).

BC1F

Sintassi: `BC1F cc, offset`
Descrizione: `if FCSR_FCC[cc] == 0 then branch`
Verifica se il valore booleano FCSR_FCC[cc] è falso ed effettua, in tal caso, un salto PC-relative. Questa istruzione può accedere al registro FCSR solo in lettura; l'informazione dev'essere scritta da una precedente istruzione condizionale del tipo `C.condizione.D`.
Esempio:
```
C.EQ.D 7,f1,f2
BC1F 7,label
```
In questo esempio `C.EQ.D` verifica l'uguaglianza tra i registri f1 ed f2, scrivendo il risultato booleano del confronto nel bit 7 del campo FCC del registro FCSR. Dopodichè `BC1F` verifica se quel bit vale 0 (falso) e salta a label.

BC1T

Sintassi: `BC1T cc, offset`
Descrizione: `if FCSR_FCC[cc] == 1 then branch`
Verifica se il valore booleano FCSR_FCC[cc] è vero ed effettua, in tal caso, un salto PC-relative. Questa istruzione può accedere al registro FCSR solo in lettura; l'informazione dev'essere scritta da una precedente istruzione condizionale del tipo `C.condizione.D`.
Esempio:
```
C.EQ.D 7,f1,f2
BC1T 7,label
```
In questo esempio, `C.EQ.D` verifica l'uguaglianza tra i registri f1 ed f2, scrivendo il risultato booleano del confronto nel bit 7 del campo FCC del registro FCSR. Dopodichè `BC1T` verifica se quel bit vale 1 (vero) e salta a label.

C.EQ.D

Sintassi: `C.EQ.D cc, fs, ft`

Descrizione: `FCSR_FCC[cc] = (fs==ft)`
Verifica il predicato "uguale a" tra i due registri fs ed ft e salva il risultato booleano in FCSR_FCC[cc]. Questo valore verrà utilizzato da un'istruzione successiva per effettuare un salto condizionato (branch) o un movimento di dati condizionato tra registri floating point.

Esempio:
```
C.EQ.D 2,f1,f2
MOVT.D f8,f9,2
```
In questo esempio `C.EQ.D` verifica l'uguaglianza tra i registri f1 ed f2, scrivendo il risultato booleano del confronto nel bit 2 del campo FCC del registro FCSR. Dopodichè `MOVT.D` verifica se quel bit vale 1 (vero), e copia il registro f9 su f8.

Eccezioni: Invalid Operation è lanciata quando fs o ft contengono valori QNaN (se attiva, si ha una trap) o SNaN(si ha sempre una trap).

C.LT.D

Sintassi: `C.LT.D cc, fs, ft`

Descrizione: `FCSR_FCC[cc] = (fs<ft)`
Verifica il predicato "minore di"(Less Than) tra i due registri fs ed ft, e salva il risultato booleano in FCSR_FCC[cc]. Questo valore verrà utilizzato da un'istruzione successiva, per effettuare un salto condizionato (branch), o per un movimento di dati condizionato tra registri floating point.

Esempio:
```
C.LT.D 2,f1,f2
BC1T 2,target
```
In questo esempio, `C.LT.D` verifica se f1 è minore di f2, scrivendo il risultato booleano del confronto nel bit 2 del campo FCC del registro FCSR. Dopodichè, `BC1T` verifica se quel bit vale 1 (vero), e salta a `target`

Eccezioni: Invalid Operation è lanciata quando fs o ft contengono valori QNaN (se attiva, si ha una trap) o SNaN(si ha sempre una trap).

CVT.D.L

Sintassi: `CVT.D.L fd, fs`

Descrizione: `fd = convert_longToDouble(fs)`
Converte un long in un double

Esempio:
```
DMTC1 r6,f5
CVT.D.L f5,f5
```
In questo esempio, `DMTC1` copia il valore del GPR r6 nell'FPR f5. Successivamente, `CVT.D.L` converte il numero in f5 da long a double.
Supponendo $r6 = 52$, dopo l'esecuzione di `DMTC1`, l'equivalente binario di 52 viene copiato nel registro f5 (f5 non contiene ancora il valore 52.0 perchè non è stato formattato ancora come double). Dopo l'esecuzione di `CVT.D.L`, $f5 = 52.0$.

Eccezioni: Invalid Operation è lanciata quando fs contiene QNaN,SNaN o Infinito

CVT.D.W

Sintassi: `CVT.D.W fd, fs`

Descrizione: `fd = convert_IntToDouble(fs)`
Converte un int in un double

Esempio:
```
MTC1 r6,f5
CVT.D.W f5,f5
```
In questo esempio, `MTC1` copia i 32 bit più bassi del GPR r6 nell'FPR f5. Successivamente, `CVT.D.W`, leggendo prima f5 come int, lo sovrascrive in double.
Supponendo `r6=0xAAAAAAAABBBBBBBB`, dopo l'esecuzione di `MTC1` si ha che `f5=0xXXXXXXXXBBBBBBBB`; si noti che i suoi 32 bit più alti (`XX..X`) sono UNDEFINED (non sono stati sovrascritti). `CVT.D.W` legge f5 come int (`f5=-1145324613`), formattandolo poi in double (`f5=0xC1D1111111400000 =-1.145324613E9`).

Eccezioni: Invalid Operation è lanciata quando fs contiene QNaN,SNaN o Infinito

CVT.L.D

Sintassi: `CVT.L.D fd, fs`

Descrizione: `fd = convert_doubleToLong(fs, CurrentRoundingMode)`
Converte, dapprima arrotondandolo, un double in un long

Esempio:
```
CVT.L.D f5,f5
DMFC1 r6,f5
```
In questo esempio, `CVT.L.D` converte il double in f5 in un long. Dopodichè, `DMFC1` copia l'FPR f5 nel GPR r6. Il risultato di questa istruzione dipende dalla modalità di arrotondamento corrente, che viene impostata dalla scheda *Arrotondamenti FPU* della finestra *Configura* → *Impostazioni*, come in Figura 1.5.

Eccezioni: Invalid Operation è lanciata quando fs vale Infinito, XNaN, o il risultato è fuori dall'intervallo dei long $[-2^{63}, 2^{63} - 1]$

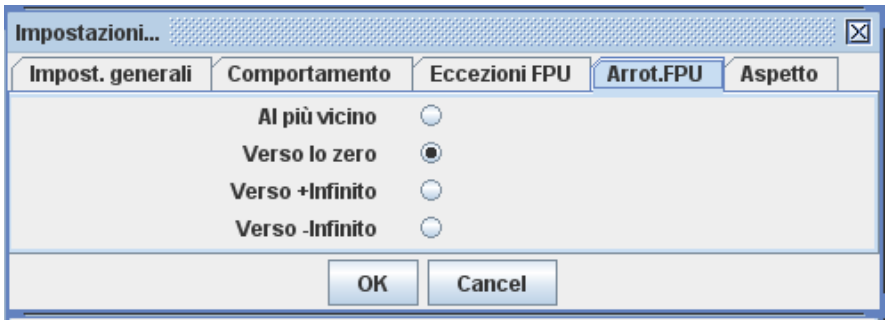

Figura 1.5: Configurazione modalità di arrotondamento

Tipo	Campo RM	Registro f5	Registro r6
Al più vicino	0	6.4	6
Al più vicino	0	6.8	7
Al più vicino	0	6.5	6 (al pari)
Al più vicino	0	7.5	8 (al pari)
Verso lo zero	1	7.1	7
Verso lo zero	1	-2.3	-2
Verso $+\infty$	2	4.2	5
Verso $+\infty$	2	-3.9	-3
Verso $-\infty$	3	4.2	4
Verso $-\infty$	3	-3.9	-4

Tabella 1.1: Esempi sui tipi di arrotondamento

CVT.W.D

Sintassi: `CVT.W.D fd, fs`

Descrizione: `fd = convert_DoubleToInt(fs, CurrentRoundingMode)`
Converte un double in un int utilizzando la modalità di arrotondamento corrente, illustrata per l'istruzione `CVT.L.D`

Eccezioni: Invalid Operation è lanciata quando fs è Infinito, XNaN, o il risultato è fuori dall'intervallo degli interi con segno $[-2^{31}, 2^{31} - 1]$

DIV.D

Sintassi: `DIV.D fd, fs, ft`

Descrizione: `fd = fs ÷ ft`

Eccezioni: Le trap di Overflow e Underflow vengono generate se il risultato non può essere rappresentato secondo lo standard IEEE 754. Invalid Operation è generata se fs o ft contengono QNaN o SNaN, o se viene eseguita un'operazione non valida ($0 \div 0$,$\infty \div \infty$). Divide by zero è generata se è eseguita una divisione per zero che non ha per dividendo un XNaN ($5 \div 0$).

DMFC1

Sintassi: `DMFC1 rt,fs`

Descrizione: `rt = fs`
Copia l'intero contenuto binario dell'FPR fs nel GPR rt. Nessun controllo di formato viene eseguito su fs prima della copia.

DMTC1

Sintassi: `DMTC1 rt, fs`

Descrizione: `fs = rt`
Copia il contenuto binario del GPR rt nell' FPR fs.

L.D ("DEPRECATED": retrocompabitibilità con WinMIPS64)

Sintassi: `L.D ft, offset(base)`
Descrizione: `memory[GPR[base] + offset]`
Carica una doubleword (64 bit) dalla memoria all'FPR ft. Questa istruzione non appartiene all'ISA MIPS64; si consiglia l'utilizzo di `LDC1`.

LDC1

Sintassi: `LDC1 ft, offset(base)`
Descrizione: `memory[GPR[base] + offset]`
Carica una doubleword (64 bit) dalla memoria all'FPR ft

LWC1

Sintassi: `LWC1 ft, offset(base)`
Descrizione: `memory[GPR[base] + offset]`
Carica una doubleword (64 bit) dalla memoria all'FPR ft

MFC1

Sintassi: `MFC1 rt, fs`
Descrizione: `rt = readInt(fs)`
Legge l'FPR fs come int e scrive il GPR rt come long
Esempio:
```
MFC1 r6,f5
SD r6,inmemoria(R0)
```
Si supponga `f5=0xAAAAAAAABBBBBBBB`; `MFC1` legge f5 come int, cioè i 32 bit più bassi (interpreta `BBBBBBBB` come `-1145324613`), e lo scrive in r6 (64 bit). Dopo l'esecuzione di `MFC1`, `r6=0xFFFFFFFFBBBBBBBB`, che equivale a `-1145324613` leggendo questo registro come long. Quindi in memoria, pur utilizzando l'istruzione SD, verrà scritta una doubleword con valore `-1145324613`. Questa operazione di conversione è chiamata *estensione del segno*, il cui approfondimento esula dagli scopi di questo manuale.

MOVF.D

Sintassi: `MOVF.D fd, fs, cc`
Descrizione: `if FCSR_FCC[cc] == 0 then fd=fs`
Verifica se la condizione di confronto booleana in FCSR_FCC[cc] è falsa e copia fs su fd. Nessun controllo sul formato viene realizzato su fs.

MOVT.D

Sintassi: `MOVT.D fd, fs, cc`
Descrizione: `if FCSR_FCC[cc] == 1 then fd=fs`
Verifica se la condizione di confronto booleana in FCSR_FCC[cc] è vera, e copia fs su fd. Nessun controllo sul formato viene realizzato su fs.

MOV.D

Sintassi: `MOV.D fd, fs`
Descrizione: `fd = fs`
Copia fs su fd senza alcun controllo del formato di fs

MOVN.D

Sintassi: `MOVN.D fd, fs, rt`
Descrizione: `if rt != 0 then fd=fs`
Copia fs su fd, senza alcun controllo del formato di fs, se il GPR rt è diverso da zero

MOVZ.D

Sintassi: `MOVZ.D fd, fs, rt`
Descrizione: `if rt == 0 then fd=fs`
Copia fs su fd, senza alcun controllo del formato di fs, se il il GPR rt è uguale a zero

MTC1

Sintassi: `MTC1 rt, fs`
Descrizione: `fs` = $\texttt{rt}_{0..31}$
Copia la word più bassa di rt scrivendola sull'FPR fs
Esempio: `MTC1 r6,f5`
Si supponga `r5=0xAAAAAAAABBBBBBBB`; `MTC1` legge i 32 bit più bassi di r5 copiandoli nei 32 bit più bassi di f5. Dopo l'esecuzione di `MTC1`, `f5=0xXXXXXXXXBBBBBBBB`; si noti che i suoi 32 bit più alti (`XX..X`) sono UNDEFINED (non sono stati sovrascritti).

MUL.D

Sintassi: `MUL.D fd, fs, ft`
Descrizione: `fd = fs`$\times$`ft`
Eccezioni: Overflow e Underflow vengono generati se il risultato non può essere rappresentato secondo lo standard IEEE754. Invalid Operation è generata se fs o ft contiene QNaN o SNaN, o se si effettua un'operazione non valida ($0 \times \infty$, QNaN$\times$numero)

S.D ("DEPRECATED": retrocompabitibilità con WinMIPS64)

Sintassi: `S.D ft, offset(base)`
Descrizione: `memory[base+ offset] = ft`
Copia la doubleword (64 bit) dell'FPR ft in memoria.

SDC1

Sintassi: `SDC1 ft, offset(base)`
Descrizione: `memory[base+ offset] = ft`
Salva la doubleword (64 bit) dell'FPR ft in memoria.

SUB.D

Sintassi: `SUB.D fd, fs, ft`
Descrizione: `fd = fs-ft`
Eccezioni: Overflow and Underflow vengono generati se il risultato non può essere rappresentato secondo lo standard IEEE754. Invalid Operation è generata se fs o ft contengono QNaN o SNaN, o se viene eseguita un'operazione non valida ($\infty - \infty$).

SWC1

Sintassi: `SWC1 ft, offset(base)`
Descrizione: `memory[base+ offset] = ft`
Salva la word (32 bit) dell'FPR ft in memoria.