Informe de la Pràctica de Fundaments de Computadors

Nom de l'estudiant:
NIU:
Assignatura: Fundaments de Computadors
Pràctica: Conversió de codi C a ARM - Anàlisi i processament de temperatures
Data de lliurament: 22 d'abril de 2025

Índex

1_celsfahr_E9M22

1.1 Especificació de les funcions bàsiques E9M22

1.1.1 E9M22_add

Aquesta funció rep dos nombres en format E9M22 (valors de 32 bits amb 1 bit de signe, 9 bits d'exponent i 22 bits de mantissa), i retorna la seva suma també en format E9M22. Es fa una normalització prèvia per alinear els exponents, es sumen les mantisses i posteriorment es renormalitza el resultat.

- **Paràmetres**:
    - `r0`: operand A (E9M22)
    - `r1`: operand B (E9M22)
- **Retorn**:
    - `r0`: resultat de la suma (E9M22)

1.1.2 E9M22_sub

Realitza la resta entre dos valors E9M22. Internament, inverteix el signe del segon operand i reutilitza l'algorisme de la suma (`E9M22_add`).

- **Paràmetres**:
  - `r0`: operand A (E9M22)
  - `r1`: operand B (E9M22)
- **Retorn**:
  - `r0`: resultat A - B en format E9M22

1.1.3 E9M22_mul

Multiplica dos valors E9M22. La funció calcula el signe resultant, suma els exponents, i multiplica les mantisses amb el desplaçament adequat per mantenir la precisió. El resultat es normalitza.

- **Paràmetres**:
  - `r0`: operand A (E9M22)
  - `r1`: operand B (E9M22)
- **Retorn**:
  - `r0`: resultat de la multiplicació (E9M22)

1.1.4 E9M22_normalize_and_round_s

Aquesta funció ajusta un valor E9M22 perquè estigui normalitzat (el bit més significatiu de la mantissa és 1) i realitza l'arrodoniment segons els bits descartats.

- **Paràmetres**:
  - `r0`: mantissa no normalitzada
  - `r1`: exponent associat
- **Retorn**:
  - `r0`: valor normalitzat i arrodonit (E9M22)

1.1.5 count_trailing_zeros_s

Compta quants bits a zero hi ha consecutius des del **bit menys significatiu cap amunt** dins d'un registre de 32 bits.

- **Paràmetres**:
  - `r0`: valor d'entrada
- **Retorn**:
  - `r0`: nombre de zeros al final

1.1.6 count_leading_zeros_s

Compta quants bits a zero hi ha consecutius des del **bit més significatiu cap avall** dins d'un registre de 32 bits.

- **Paràmetres**:
  - `r0`: valor d'entrada
- **Retorn**:
  - `r0`: nombre de zeros al començament

1.1.7 E9M22_neg

Canvia el signe d'un valor en format E9M22. Només cal invertir el bit de signe.

- **Paràmetres**:
  - `r0`: valor original (E9M22)
- **Retorn**:
  - `r0`: valor amb signe invertit

1.1.8 E9M22_abs

Retorna el valor absolut d'un nombre E9M22. Elimina el signe (posa el bit de signe a 0).

- **Paràmetres**:
  - `r0`: valor original (E9M22)
- **Retorn**:
  - `r0`: valor absolut

1.1.9 E9M22_are_eq_s

Comprova si dos valors E9M22 són iguals. Té en compte el cas especial de NaN, que mai és igual a res (ni tan sols a ell mateix).

- **Paràmetres**:
  - `r0`: operand A
  - `r1`: operand B
- **Retorn**:
  - `r0`: 1 si són iguals, 0 altrament

1.1.10 E9M22_are_unordered_s

Retorna 1 si algun dels dos operands és NaN. Aquest cas indica que no es poden comparar (unordered en terminologia IEEE-754).

- **Paràmetres**:
  - `r0`: operand A
  - `r1`: operand B
- **Retorn**:
  - `r0`: 1 si algun operand és NaN, 0 altrament

1.2 Disseny de les funcions bàsiques E9M22

1.2.1 E9M22_add

El disseny segueix l'algorisme clàssic de suma de nombres en coma flotant:

1. **Extracció de camps**: Es separen el signe, exponent i mantissa dels dos operandes.
2. **Alineació d'exponents**: Si els exponents són diferents, es desplaça cap a la dreta la mantissa del nombre amb exponent més petit, per fer coincidir els exponents.
3. **Operació de mantisses**: Si els signes són iguals, es sumen les mantisses. Si són diferents, es resten, i es conserva el signe del nombre més gran.
4. **Normalització**: Es desplaça la mantissa resultant cap a l'esquerra (si hi ha zeros a l'inici) i s'ajusta l'exponent en conseqüència.
5. **Arrodoniment**: Es tenen en compte els bits descartats per aplicar arrodoniment correcte.
6. **Recomposició**: Finalment es torna a empaquetar el signe, l'exponent i la mantissa en format E9M22.

1.2.2 E9M22_sub

Es basa en la suma, però amb el signe del segon operando invertit:

- Es canvia el bit de signe del segon operando (`B`), convertint la resta en una suma: `A - B = A + (-B)`.
- A continuació, es reutilitza el mateix algoritme de `E9M22_add`.

1.2.3 E9M22_mul

Per multiplicar dos nombres E9M22:

1. **Extracció de camps**: Signe, exponent i mantissa dels dos operandes.
2. **Càlcul del signe final**: S'usa XOR entre els signes.
3. **Exponent resultant**: Es sumen els exponents i es resta el bias (511).
4. **Multiplicació de mantisses**: S'inclou el bit implícit (1) abans de multiplicar les mantisses.
5. **Normalització**: Es fa un desplaçament si cal, i es corregeix l'exponent.
6. **Arrodoniment i saturació**: Si es sobrepassa el rang, es genera +∞ o -∞.
7. **Recomposició**: Es generen els bits finals del format E9M22.

1.2.4 E9M22_normalize_and_round_s

Aquesta funció ajusta una mantissa i exponent després d'una operació:

1. **Normalització**: Es detecta el primer bit 1 per saber quants llocs s'ha de desplaçar la mantissa.
2. **Ajust de l'exponent**: Cada desplaçament a l'esquerra incrementa l'exponent.
3. **Arrodoniment**: Es fa servir l'últim bit descartat i el següent per decidir si s'arrodoneix cap amunt.
4. **Tractament de casos límit**: Es gestiona el desbordament d'exponent i la generació de zero.

1.2.5 count_trailing_zeros_s

Implementa un algorisme per comptar zeros consecutius des del bit menys significatiu (LSB) cap a l'esquerra:

- Es fa un bucle que desplaça cap a la dreta fins que troba un 1.
- Alternativament, es pot usar `rbit` (invertir bits) i `clz` (comptar zeros des del MSB).

1.2.6 count_leading_zeros_s

Utilitza la instrucció `clz` per comptar quants zeros hi ha des del MSB fins al primer bit 1:

- És útil per normalitzar mantisses.
- Si no es pot usar `clz`, es pot fer un bucle que desplaça cap a l'esquerra i compta fins trobar un 1.

1.2.7 E9M22_neg

Inverteix el signe d'un nombre E9M22:

- Es fa un XOR amb `0x80000000`, que només modifica el bit més significatiu (bit 31).
- La resta del valor roman intacta.

1.2.8 E9M22_abs

Retorna el valor absolut:

- Es posa el bit de signe a 0 fent un AND amb `0x7FFFFFFF`.
- Això manté l'exponent i la mantissa intactes.

1.2.9 E9M22_are_eq_s

Compara si dos nombres E9M22 són iguals:

1. **Gestió de NaN**: Si qualsevol operand és NaN, retorna 0.
2. **Comparació binària**: Si cap és NaN, es comparen directament tots els bits.

1.2.10 E9M22_are_unordered_s

Determina si qualsevol dels dos operands és NaN:

- Comprova si l'exponent és tot uns (`0x1FF`) i la mantissa no és zero.
- Si algun operand és NaN, retorna 1 (estan "desordenats" i no comparables).

1.3 Implementació

Fragments de codi ARM rellevants amb comentaris. Explica:

- Com s'ha traduït el codi C a codi ARM
- Com s'han fet les operacions aritmètiques i lògiques
- Com s'ha controlat el flux i gestionat els registres

1.1.1 E9M22_add

1.1.2 E9M22_sub

1.1.3 E9M22_mul

1.1.4 E9M22_normalize_and_round_s

1.1.5 count_trailing_zeros_s

1.1.6 count_leading_zeros_s

```
.global count_trailing_zeros_s
count_trailing_zeros_s:
    push {lr}              @ Guardem l'adreça de retorn a la pila
    mov r1, #0             @ Inicialitzem el comptador de zeros a r1

.loop_ctz:
    tst r0, #1             @ Comprovem si el bit menys significatiu és 1
    bne .end_ctz           @ Si ho és, sortim del bucle
    lsr r0, r0, #1         @ Desplacem r0 un bit cap a la dreta (r0 = r0
>> 1)
    add r1, r1, #1         @ Incrementem el comptador de zeros finals
    b .loop_ctz            @ Tornem a comprovar el següent bit

.end_ctz:
    mov r0, r1             @ El resultat (comptador) el posem a r0
    pop {pc}               @ Recuperem l'adreça de retorn i retornem
```

1.1.7 E9M22_neg

1.1.8 E9M22_abs

1.1.9 E9M22_are_eq_s

1.1.10 E9M22_are_unordered_s