Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego

Laboratorium z przedmiotu:
Architektura i organizacja komputerów

Sprawozdanie z ćwiczenia laboratoryjnego nr 8:

Hazardy sterowania w przetwarzaniu potokowym

Spis treści

A. Treść zadania

Begin

Dane:

SKŁADNIK = 1330, UŁAMEK = 0.33, ROZMIAR = 103.

Wzór:

TB[i] = [(3.2*T[i] * T[i+3]*T[i+5]* T[i+6])- T[i+7]] / [T[i+3]]

Napisać program ***Lab8_nr.s*** w asemblerze komputera DLX, który:

1. Zadeklaruje dwie tablice przechowujące liczby zmiennoprzecinkowe podwójnej precyzji: ***T*** 130-elementową oraz ***TB*** ROZMIAR-elementową, a także zmienną ***Suma*** zmiennoprzecinkową podwójnej precyzji.
2. Komórki tablicy ***T*** wypełni (za pomocą obliczeń, wykonanych w pętli, a nie za pomocą statycznej deklaracji z nadaniem wartości początkowych) kolejnymi liczbami o części ułamkowej równej UŁAMEK i części całkowitej rosnącej o jeden, począwszy od numeru w dzienniku studenta/ studentki, powiększonej o SKŁADNIK
   (np. nr=1; UŁAMEK = 0.35; SKŁADNIK = 5; w tablicy T mają być zapisane liczby T[1] = (1+5+0.35) = 6.35, T[2] = (6.35 + 1) = 7.35 itd.).
3. Następnie dla każdego elementu tablicy ***TB*** wykona operację, określoną powyższym wzorem (UWAGA: wszystkie występujące we wzorze działania mają być jawnie wykonane w programie, nie są dopuszczalne przekształcenia wzoru (np. skrócenia), zastępowanie wykonywania działań obliczonymi stałymi. Można użyć stałych dla reprezentowania w programie wartości numeru w dzienniku, danych SKŁADNIK i UŁAMEK oraz stałych we wzorach na TB np. 3.2 itd.
4. W zmiennej ***Suma*** umieści obliczoną w pętli sumę wszystkich elementów tablicy ***TB***. Uwaga - ze względu na błąd w implementacji forwardingu ZMP
   w WinDLX czasem zdarza się tak, że poprawnie napisany program przy wyłączonym forwardingu "daje" poprawne wyniki, a po włączeniu forwardingu generuje złe zawartości TB albo błędną Sumę. Radzę w przypadku "niezrozumiałych" błędów wyłączyć forwarding i sprawdzić działanie programu. Szczegóły wspomnianego błędu można poznać tutaj.
5. Przed rozpoczęciem tworzenia programu radzę (o ile Studentka/Student - wykonawca ćwiczenia walczy o ocenę co najmniej **db**) zaprojektować arkusz kalkulacyjny w Excelu, Calcu lub innym środowisku, wykonujący te same obliczenia w celu weryfikacji poprawności uzyskiwanych w programie wyników. Arkusz zapisać do postaci .xls albo .xlsx.

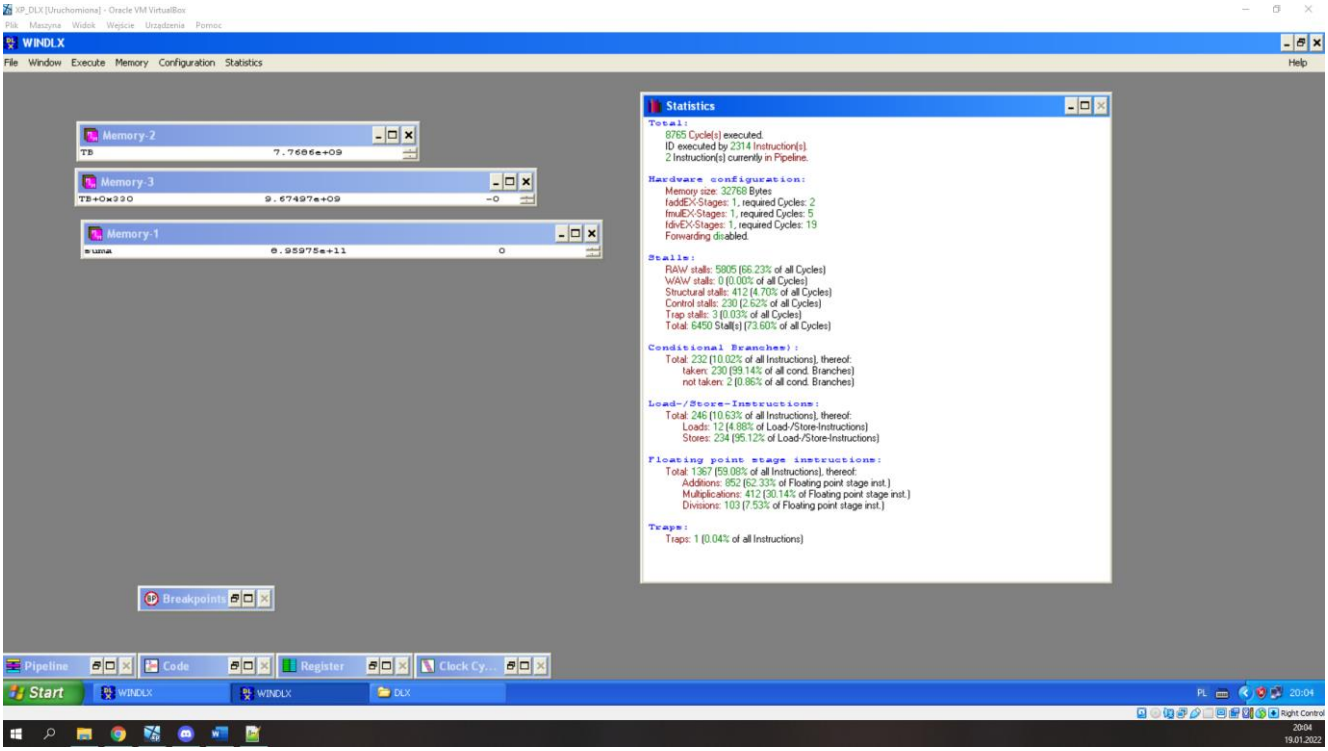
End

B. Kod programu

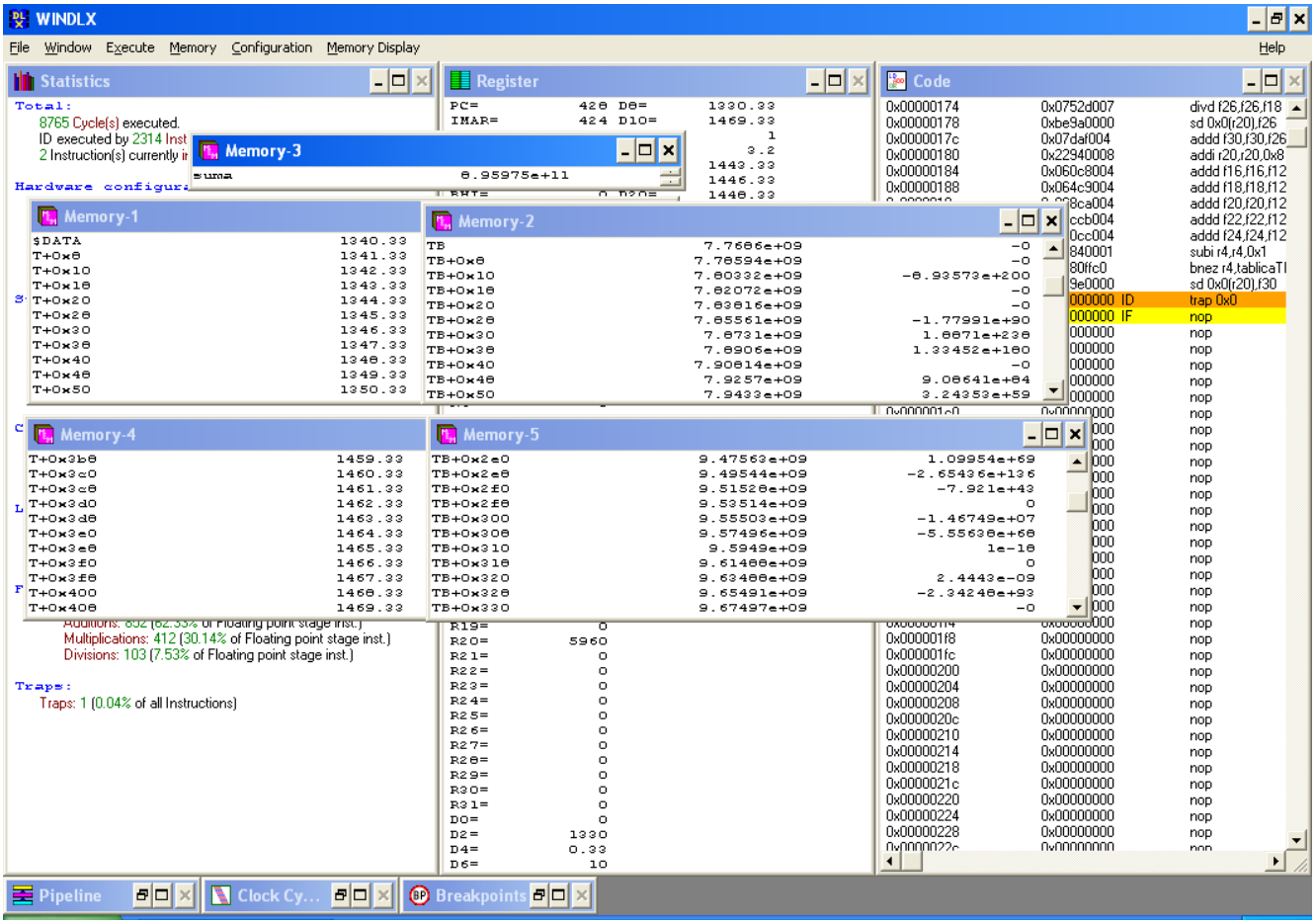
```
.data
T: .space 1040
TB: .space 824
suma: .double 0

skladnik: .double 1330
ulamek: .double 0.33
numer: .double 10

stala: .double 3.2
liczT: .word 129
liczTB: .word 103
jeden: .double 1

.text
;pobieranie danych
ld f2, skladnik     ; stala skladnik
ld f4, ulamek       ; stala ulamek
ld f6, numer        ; stala numer
ld f12, jeden       ; liczba 1 typu double
ld f14, stala       ; stala do obliczen dla TB

;indeks i licznik dla T
lw r10, liczT       ; pobierz licznik T
addi r20, r0, T     ; indeks pierwszego elementu T

;pierwszy element T
addd f8, f2, f4     ; skladnik + ulamek
addd f10, f8, f6    ; skladnik + ulamek + numer
sd T(r0), f10       ; wpisz wynik do T[1]

;wypelnianie tabliy T
tablicaT:
addi r20, r20, #8   ; miejsce nastepnego elementu T (8 = sizeof(double))
addd f10, f10, f12  ; nastepna wartosc T
sd 0(r20), f10      ; wpisz do T[i]
subi r10, r10, #1   ; zmniejsz licznik
bnez r10, tablicaT   ; zakoncz, jesli licznik = 0

;indeks i licznik dla TB
lw r4, liczTB       ; pobierz licznik TB
addi r20, r20, #8   ; indeks pierwszego elementu TB (T[131] => TB[1])

;przygotowywanie zmiennych do obliczen dla TB
add r5, r0, r20     ; indeks nastepnej pamieci po T
subi r5, r5, #1040  ; indeks pierwszego elementu T
ld f16, 4096        ; wartosc T[i]
```

```
ld f18, 4120        ; wartosc T[i+3]
ld f20, 4136        ; wartosc T[i+5]
ld f22, 4144        ; wartosc T[i+6]
ld f24, 4152        ; wartosc T[i+7]

;wypenianie tablicy TB
tablicaTB:
;obliczanie wartosci TB[i]
multd f26, f14, f16 ; iloczyn = stala * T[i]
multd f26, f26, f18 ; iloczyn *= T[i+3]
multd f26, f26, f20 ; iloczyn *= T[i+5]
multd f26, f26, f22 ; iloczyn *= T[i+6]
subd f26, f26, f24  ; iloczyn -= T[i+7]
divd f26, f26, f18  ; iloczyn /= T[i+3]

;dodawanie wartosci do TB[i]
sd 0(r20), f26      ; wpisz do TB[i]
addd f30, f30, f26  ; dodaj wartosc TB[i] do sumy
addi r20, r20, #8   ; nastepny indeks TB[i]

;nastepne wartosci do przeliczania
addd f16, f16, f12  ; nastepna wartosc T[i]
addd f18, f18, f12  ; nastepna wartosc T[i+3]
addd f20, f20, f12  ; nastepna wartosc T[i+5]
addd f22, f22, f12  ; nastepna wartosc T[i+6]
addd f24, f24, f12  ; nastepna wartosc T[i+7]

;zarzadzanie petla
subi r4, r4, #1     ; zmniejsz licznik
bnez r4, tablicaTB  ; zakoncz, jesli licznik = 0

;wpisywanie wyniku do zmiennej suma
sd 0(r20), f30      ; ostatnia zmiana indeksu (TB[104] => Suma)




trap 0
```

C. Zrzut ekranu z wynikami



D. Zawartość tablic T oraz TB



E. Obliczone wartości tablic T oraz TB

i	T[i]	i	TB[i]
1	1340,33	1	7768587587
2	1341,33	2	7785941199
3	1342,33	3	7803320635
4	1343,33	4	7820725914
5	1344,33	5	7838157056
6	1345,33	6	7855614079
7	1346,33	7	7873097003
8	1347,33	8	7890605846
9	1348,33	9	7908140629
10	1349,33	10	7925701370
121	1460,33	94	9495439817
122	1461,33	95	9515277192
123	1462,33	96	9535142178
124	1463,33	97	9555034792
125	1464,33	98	9574955053
126	1465,33	99	9594902983
127	1466,33	100	9614878598
128	1467,33	101	9634881919
129	1468,33	102	9654912965
130	1469,33	103	9674971754

Suma	895974923151,437
	8,96E+11

F. Pomiar liczby cykli dla danych konfiguracji

Lp.	Forwarding	l. j. z. dodających	l. j. z. mnożących	Suma	Liczba cykli
1	On	1	1	6.41082e+08	6928
2	On	1	4	6.41082e+08	6928
3	On	4	1	6.41082e+08	6516
4	On	4	4	6.41082e+08	6516
5	Off	1	1	8.9597e+11	8765
6	Off	1	4	8.9597e+11	8765
7	Off	4	1	8.9597e+11	8353
8	Off	4	4	8.9597e+11	8353

G. Konfiguracja szybka

Lp.	Forwarding	l. j. z. dodających	l. j. z. mnożących	Suma	Liczba cykli
7	Off	4	1	8.9597e+11	8353

Zrzut ekranu:

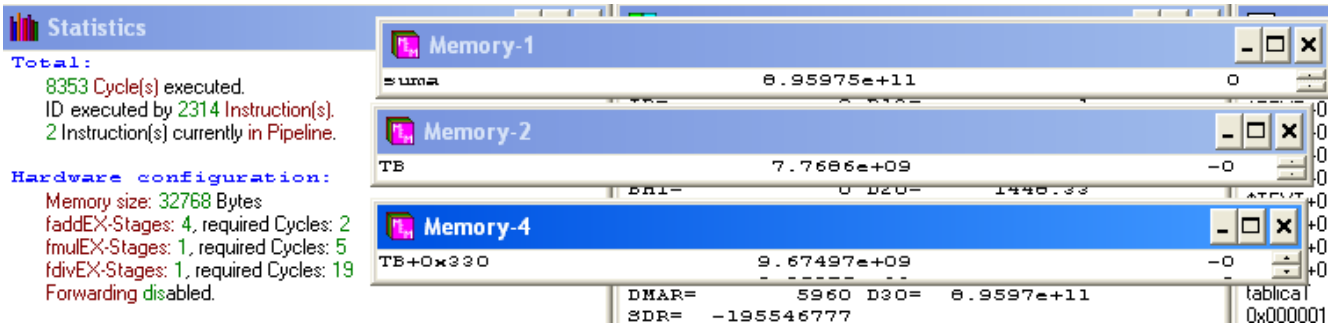
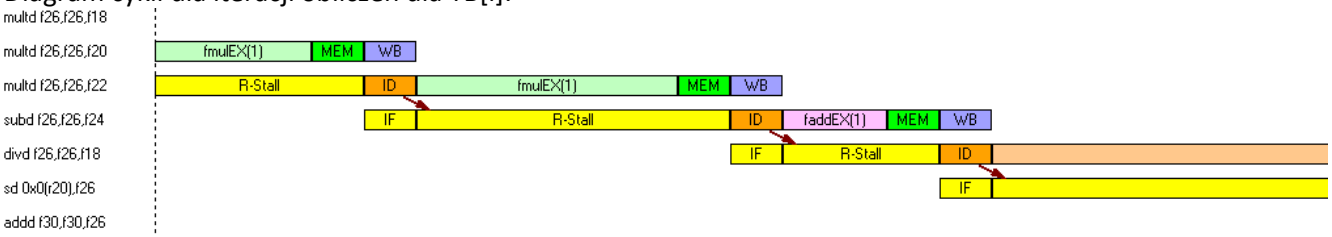


Diagram cykli dla iteracji obliczeń dla TB[i]:



Czerwona strzałka wskazuje miejsce, w którym konieczne jest zastosowanie „R-Stall". Przyczyną tego jest to, że instrukcja wskazywana przez strzałkę musi poczekać dodatkową ilość cykli, żeby poprzednia instrukcja mogła się wykonać (zjawisko hazardu).

Ta sama sytuacja dotyczy się następnego hazardu. Widać tutaj, że przez wyłączony forwarding oraz bardzo długi czas wykonania operacji „fdiv" następna instrukcja została pobrana, ale nie może się wykonać dopóki poprzednia instrukcja się nie zakończy

H. Konfiguracja wolna

Lp.	Forwarding	l. j. z. dodających	l. j. z. mnożących	Suma	Liczba cykli
5	Off	1	1	8.9597e+11	8765

Zrzut ekranu:

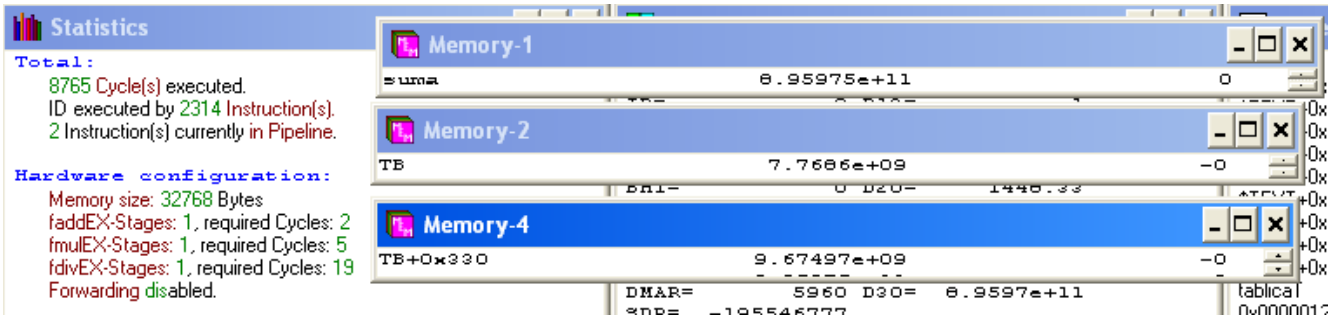
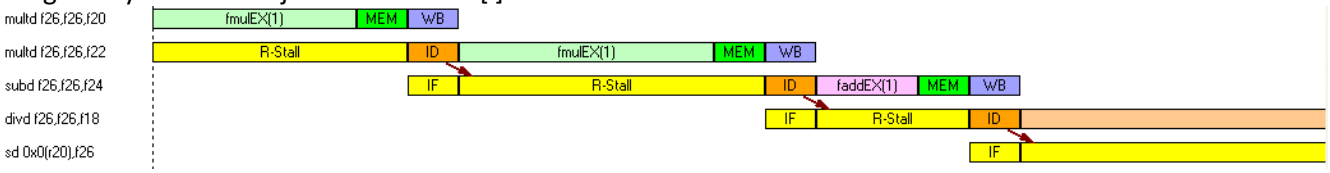
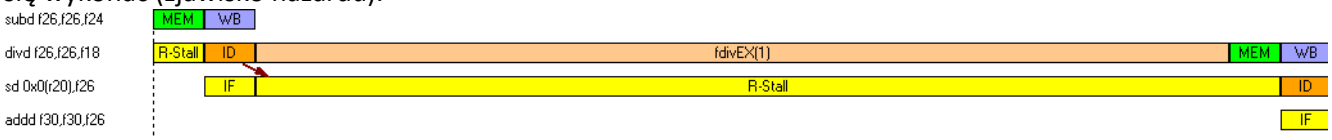


Diagram cykli dla iteracji obliczeń dla TB[i]:



Czerwona strzałka wskazuje miejsce, w którym konieczne jest zastosowanie „R-Stall”. Przyczyną tego jest to, że instrukcja wskazywana przez strzałkę musi poczekać dodatkową ilość cykli, żeby poprzednia instrukcja mogła się wykonać (zjawisko hazardu).



Ta sama sytuacja dotyczy się następnego hazardu. Widać tutaj, że przez wyłączony forwarding oraz bardzo długi czas wykonania operacji „fdiv” następna instrukcja została pobrana, ale nie może się wykonać dopóki poprzednia instrukcja się nie zakończy

I. Porównanie diagramów w/w konfiguracji oraz ich hazardów

Jednostki, które zostały ustawione służą do określania liczby interwencji między instrukcją produkującą wynik, a instrukcją pobierającą wynik. Oznacza to, że przy zaistnieniu zjawiska hazardu instrukcja będąca w stanie „R-Stall” będzie w trakcie wykonywania poprzedniej instrukcji podejmować więcej prób przejęcia wyniku. W momencie, gdy uda się jej uzyskać możliwość skorzystania z wyniku, zacznie wykonywać się szybciej, dzięki czemu zaistnieje optymalizacja i liczba cykli działania programu się zmniejszy.

J. Obliczanie CPI dla w/w konfiguracji

CPI (clock cycles per instruction) = liczba cykli / liczba wykonanych instrukcji

Konfiguracja szybka:

8353 / 2314 = 3,6097

Konfiguracja wolna:

8765 / 2314 = 3,7878