Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego

Laboratorium z przedmiotu:
Architektura i organizacja komputerów

Sprawozdanie z ćwiczenia laboratoryjnego nr 7:

Hazardy danych w przetwarzaniu potokowym

Spis treści

A. Treść zadania

Lab7 (13-14) Y4 prawdziwe
Parametrem zadania będzie **nr = numer_w_dzienniku autorki/ autora sprawozdania (numer na liście grupy w USOS).**
Begin
Napisać program **L7_nr** w asemblerze komputera WinDLX, który

1. Zadeklaruje dwie tablice przechowujące liczby całkowite:
   TA 110- elementową oraz
   TB 100-elementową
   a także zmienną zmiennoprzecinkową podwójnej precyzji .double o nazwie **Suma** i nada jej wartość początkową zero.

2. Komórki tablicy ***TA*** wypełni rosnąco kolejnymi liczbami całkowitymi
   począwszy od (1000 + nr) – dla nr = 5 będą to odpowiednio 1005, 1006, 1007 itd.

3. Następnie dla każdego elementu tablicy ***TB*** wykona operację:
   TB[i] = (TA[i+2] + TA[i+5]) * 200

4. W pętli obliczy do rejestru **Rnr** sumę elementów tablicy TB.

5. Zawartość **Rnr** zapisze po niezbędnej konwersji do zmiennej **Suma**.

End

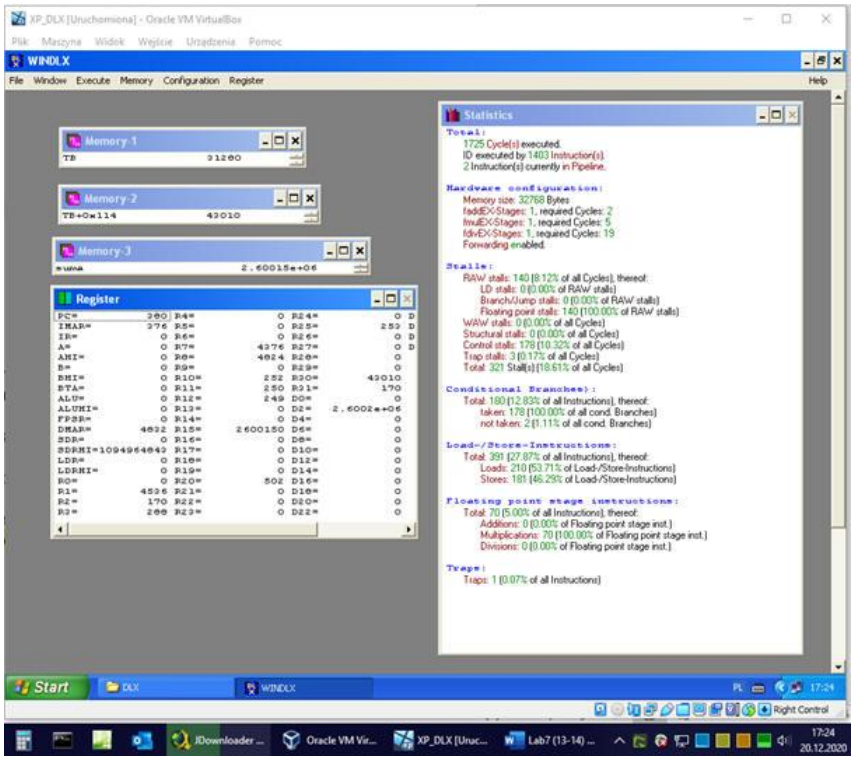
Przed zakończeniem czasu zajęć wykonać zrzut ekranowy z wynikami uruchomienia programu **L7_nr,** pokazujący po lewej stronie okna WinDLX 4 okienka podglądu:

- Pierwszego elementu **tablicy TB**
- Ostatniego elementu **tablicy TB**
- Zmiennej .double **Suma**
- Rejestrów R i D, w tym **Rnr z całkowitoliczbową** wersją **sumy elementów tablicy TB i D4 (dla całej grupy) ze zmiennoprzecinkową** wersją **sumy elementów tablicy TB.**

A po prawej stronie okna WinDLX zawartość całego okienka **Statistics**. Na zrzucie ma być widoczny **zegar systemowy** z aktualną godziną (prawy dolny róg ekranu Win 10).

Uwaga: symulacja pracy tego programu nie będzie natychmiastowa – troszkę to będzie trwało.

Przykładowy wygląd zrzutu ekranowego (suma STP - w R15, suma ZMP niepoprawnie dla Waszej grupy, ale to tylko przykład prezentacji wyniku - w D2):



W sprawozdaniu muszą się znaleźć:

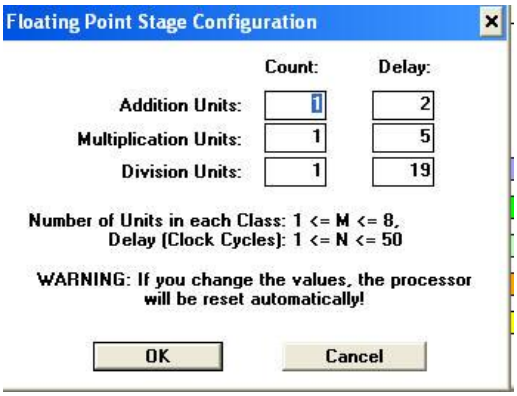
1. Strona tytułowa z nazwą przedmiotu, tematem zajęć, datą wykonania ćwiczenia, danymi wykonawcy.
2. Treść zadania z mojej strony (Begin – End).
3. Treść napisanego przez wykonawcę programu w wersji wysłanej w dniu przeprowadzenia zajęć (w postaci tekstowej, możliwej do skopiowania z pliku .pdf) i zrzut ekranu z wynikami programu w wersji wysłanej w dniu przeprowadzenia zajęć.
4. O ile zaszła konieczność modyfikacji programu dla uzyskania prawidłowych wyników - nową treść programu i nowy zrzut ekranu według wymagań z dnia zajęć.
5. Wydruk zawartości zmiennej SUMA oraz pierwszych 10 i ostatnich 10 elementów tablic TA i TB z podglądu zawartości pamięci operacyjnej WinDLX, potwierdzających poprawne wykonanie prezentowanego programu – nowe okienka, nieobecne na zrzutach ekranu, przysyłanych na koniec zajęć.

Oceny:

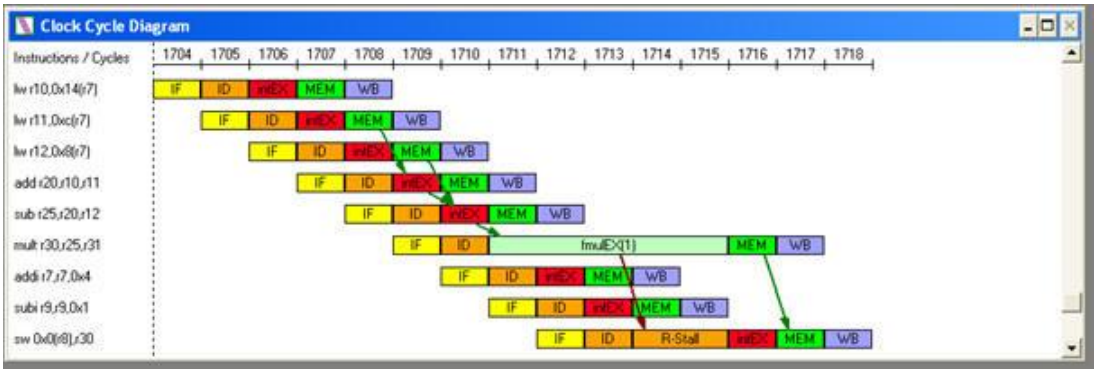
- Brak maila ze zrzutem ekranu po uruchomieniu programu i treścią programu – jako załącznikami przed końcem czasu zajęć, albo brak sprawozdania w terminie = zero do średniej.
- Dotrzymane terminy i warunki, ale program nie działa poprawnie nawet co do sumy elementów TB w Rnr na koniec czasu zajęć = **ndst**. Trzeba po zajęciach poprawić przynajmniej program w zakresie do sumy elementów TB w Rnr i przysłać w sprawozdaniu pkt. 1,2,3,4,5. Poprawienie programu po czasie zajęć nie zmienia oceny. Dalej jest ndst. Ale nie zero!
- Niepoprawnie działający program – na koniec czasu zajęć dobrze wypełnia obie tablice, liczy sumę do Rnr, ale źle konwertuje do **D4**, przy spełnieniu wymagań co do terminów, maili i załączników = **dst**.
- Wszystko w terminie czasu zajęć liczy się i prezentuje dobrze = **db**.

Aby uzyskać ocenę **bardzo dobrą**, należy ponadto w sprawozdaniu zrealizować poniższe punkty A i B w taki sposób, żeby zrozumienie przez wykonawcę prezentowanych zagadnień nie budziło moich wątpliwości 😊:

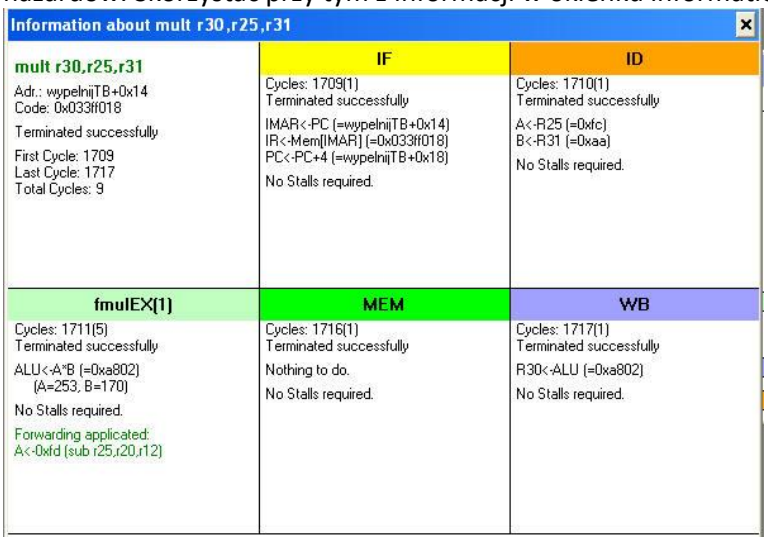
A. Zmierzyć liczbę cykli zegarowych dla wykonania uzyskanego programu przy 1) załączonym 2) wyłączonym forwardingu (WinDLX, menu Configuration/ Enable Forwarding). Wyniki pomiarów przedstawić w postaci tabeli. Dla każdego uruchomienia sprawdzić poprawność obliczonej zmiennej ***Suma***. Nie zmieniać liczby ani domyślnych czasów działania faz zmiennoprzecinkowych: menu Configuration/ Floating Point Stages



B. W sprawozdaniu zamieścić zrzut okienka Clock Cycle Diagram, z odpowiednią zawartością, podobny do poniższego



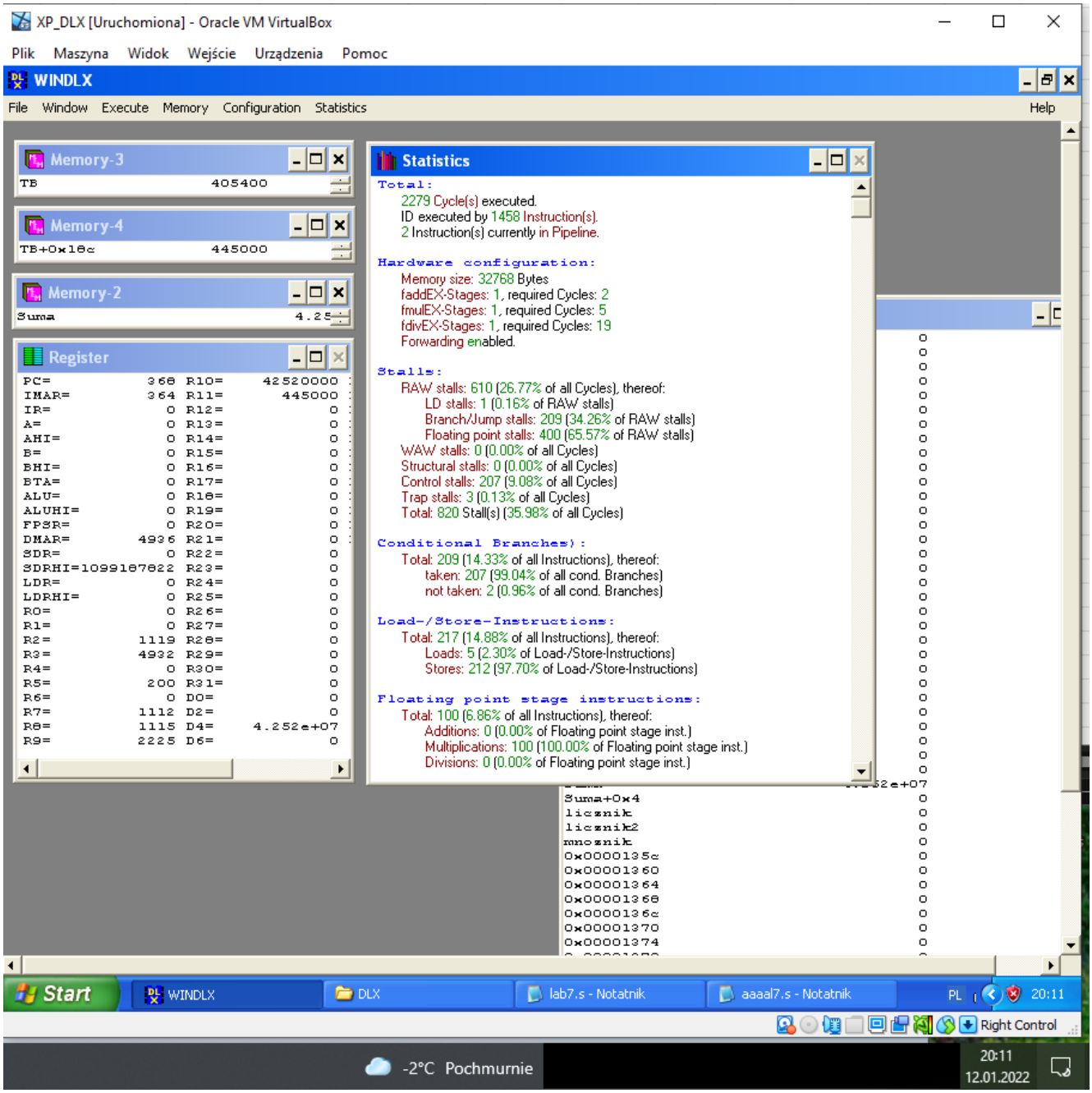
i porównać diagramy cykli zegarowych dla wykonania jednej iteracji wypełniania tablicy TB - obliczeń z punktu 3 (TB[i] = …) dla konfiguracji: ***1) Enable Forwarding off*** i ***2) Enable Forwarding on***. Opisać co najmniej 2 występujące dla wyłączonego forwardingu w tej iteracji hazardy danych i przyczyny ich powstania. Wyjaśnić wpływ załączenia forwardingu na działanie WinDLX dla tych hazardów. Skorzystać przy tym z informacji w okienku Information, na przykład:



B. Kod programu

```
.data
TA: .space 440
TB: .space 400
Suma: .double 0
licznik: .word 109
licznik2: .word 100
mnoznik: .word 200

.text
lw r1, licznik      ; pobierz licznik
addi r2, r0, #1010  ; podaj pierwsza wartosc
sw TA(r0), r2       ; wpisz pierwsza wartosc do TA
addi r3, r0, TA     ; Indeks pierwszego elementu TA

tablicaA:
addi r3, r3, #4     ; Nastepny indeks
addi r2, r2, #1     ; Nastepna wartosc
sw 0(r3), r2        ; wpisz do indeksu dana wartosc
subi r1, r1, #1     ; zmniejsz licznik
bnez r1, tablicaA   ; zakoncz, jesli licznik = 0

lw r4, licznik2     ; pobierz licznik2
lw r5, mnoznik      ; pobierz mnoznik
sw TB(r0), r11      ; pobierz indeks 1 elementu TB
lw r7, 4104         ; indeks TA[2]
lw r8, 4116         ; indeks TA[5]

tablicaB:
add r9, r7, r8      ; r9 = TA[i+2] + TA[i+5]
addi r3, r3, #4     ; nastepny indeks (TA[110] -> TB[0])
mult r11, r9, r5    ; r11 = (TA[i+2] + TA[i+5]) * 200
add r10, r10, r11   ; dodaj wartosc do sumy
sw 0(r3), r11       ; wpisz do indeksu wartosc
addi r7, r7, #1     ; nastepny indeks TA[i+2]
addi r8, r8, #1     ; nastepny indeks TA[i+5]
subi r4, r4, #1     ; zmniejsz licznik2
bnez r4, tablicaB   ; zakoncz, jesli licznik2 = 0

movi2fp F4, R10     ; pobierz jako float sume
cvti2d F4, F4       ; skonwertuj jako double
sd Suma, F4         ; wpisz do zmiennej Suma

trap 0
```

C. Zrzut ekranu z wynikami

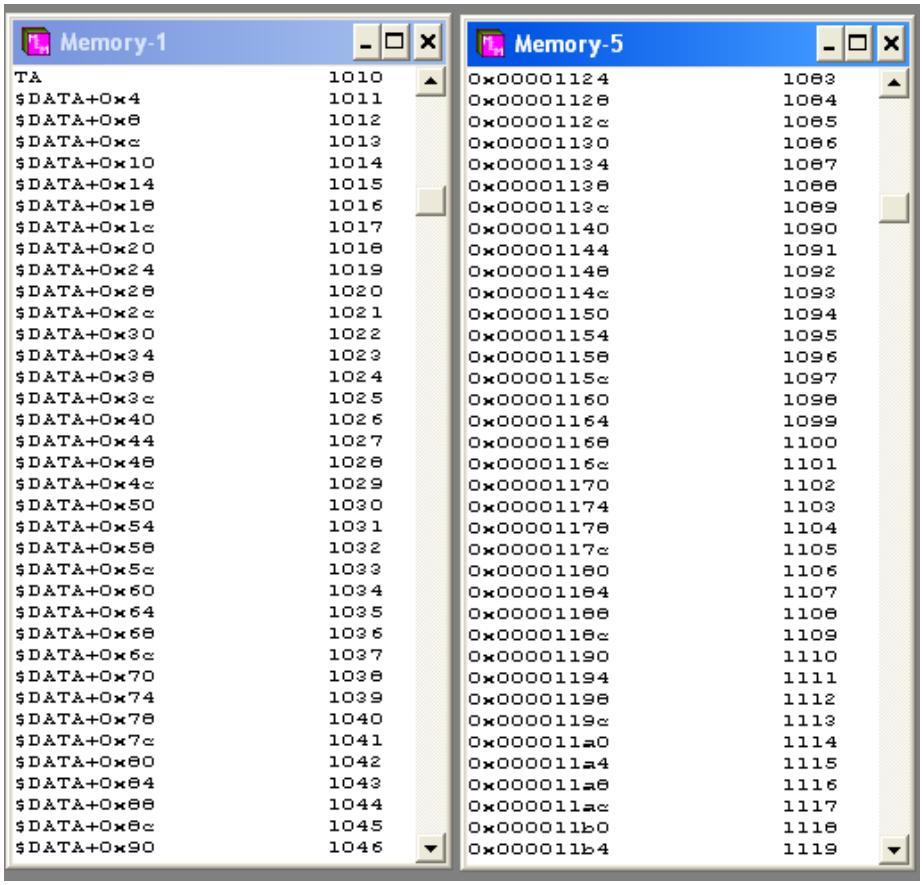


D. Zawartość tablic TA oraz TB

Zmienna SUMA

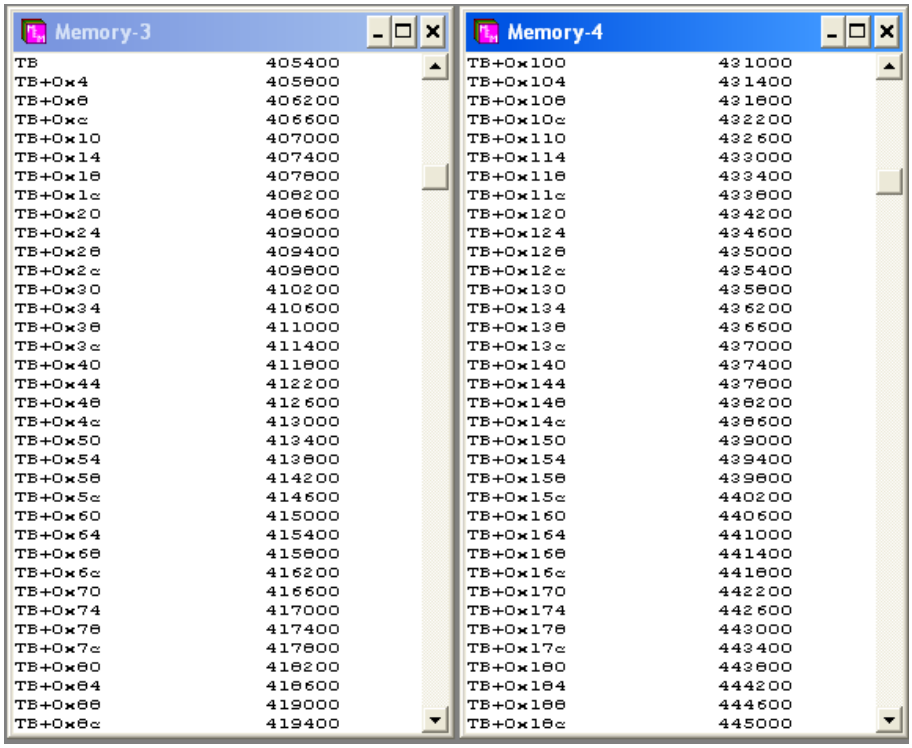
Tablica TA – pierwsze i ostatnie 10 elementów

(Po lewej – 10 od góry, po prawej – 10 od dołu)



Tablica TB – pierwsze i ostatnie 10 elementów

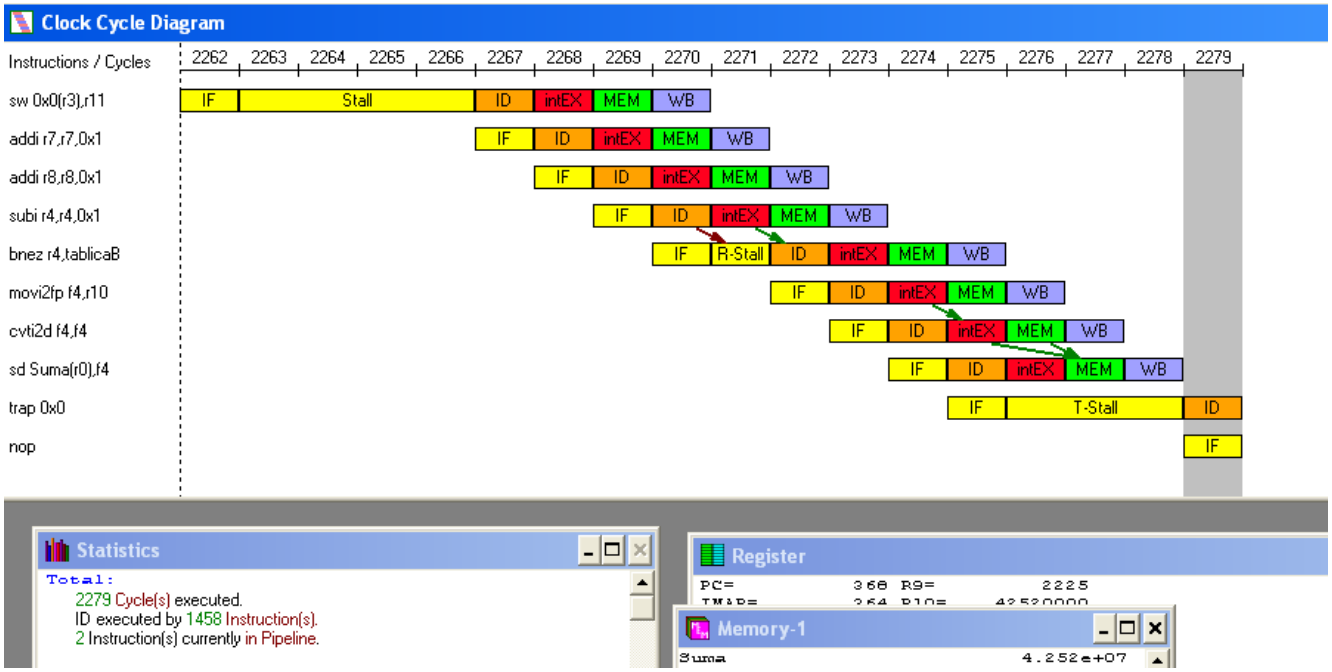
(Po lewej – 10 od góry, po prawej – 10 od dołu)



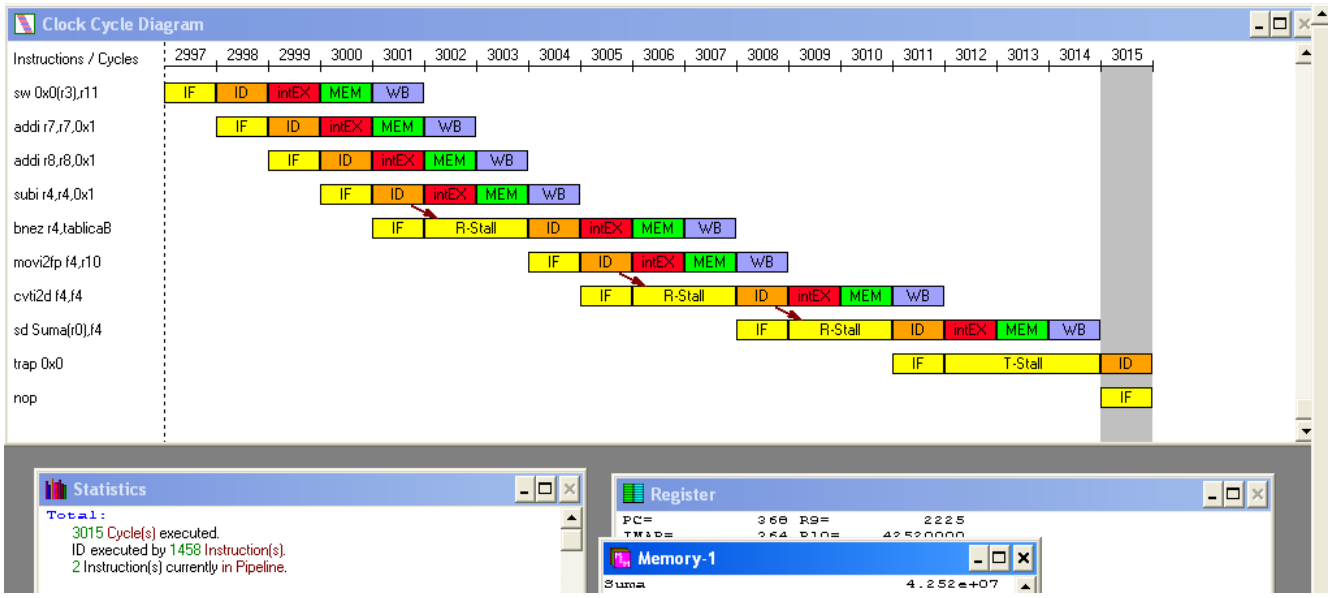
E. Pomiar liczby cykli zegarowych

Forwarding	Ilość cykli	Wartość zmiennej Suma
Włączony	2279	4.252e+07
Wyłączony	3015	4.252e+07

Włączony forwarding



Wyłączony forwarding

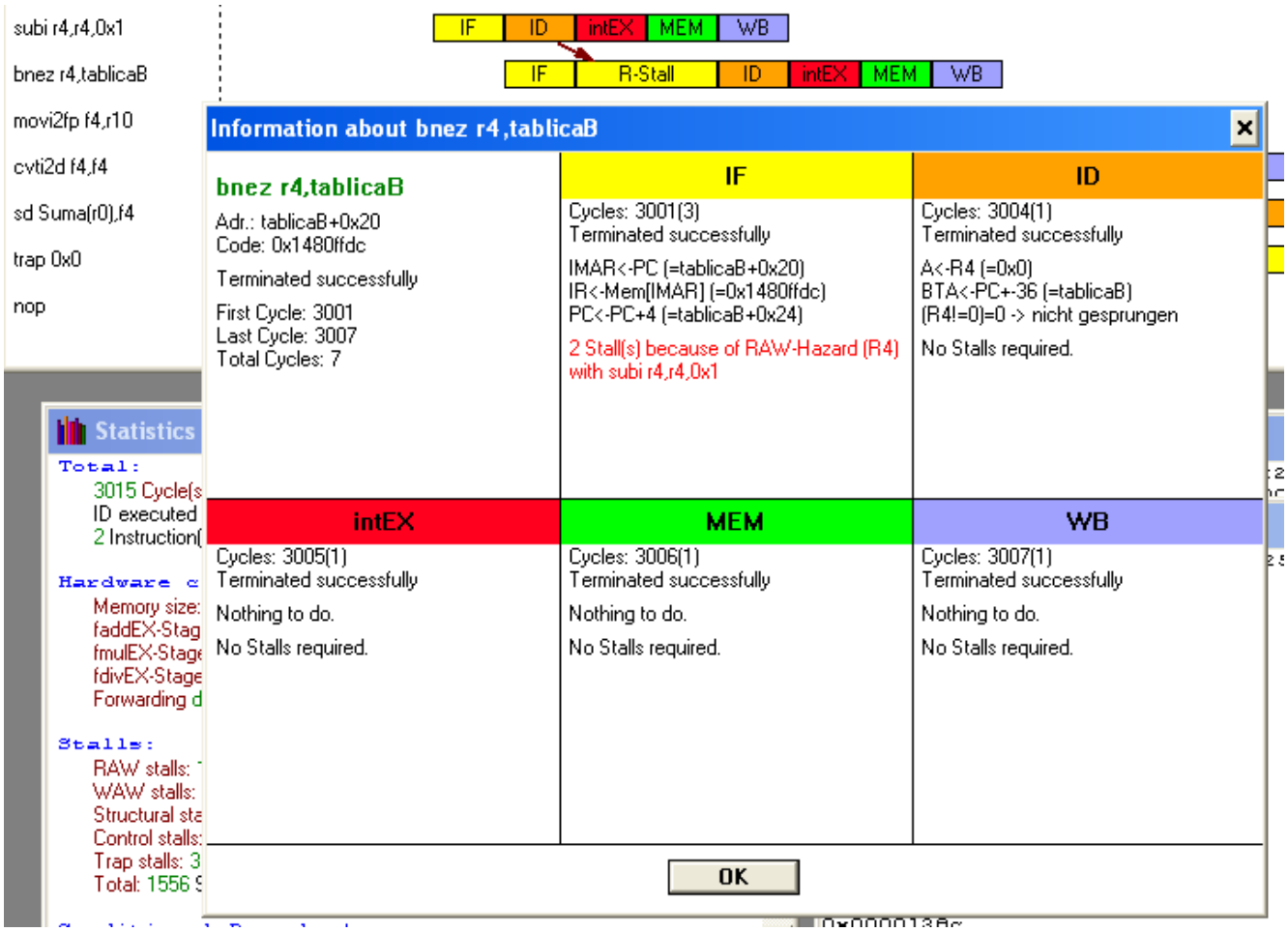


F. Opisy hazardów dla wyłączonego forwardingu

Hazard jest to zjawisko, które wynika z faktu, że nie wszystkie instrukcje są w stanie wykonać się w trakcie jednego cyklu. Przez to, jeśli instrukcje następujące po wspomnianych wykorzystują te same elementy pamięci, przez wzgląd na bezpieczeństwo zmuszone są do wstrzymania działania i oczekiwania na pełne wykonanie swojego poprzednika. Jest to sytuacja niekorzystna, prowadząca do zwiększania się liczby cykli, w których wykonuje się program.

W celu zmniejszenia ilości cykli w programie można zastosować forwarding. Zmniejsza on czas przestoju (Stall) poprzez wykorzystywanie wyników poprzedniej instrukcji.

instrukcja „bnez r4, tablicaB”

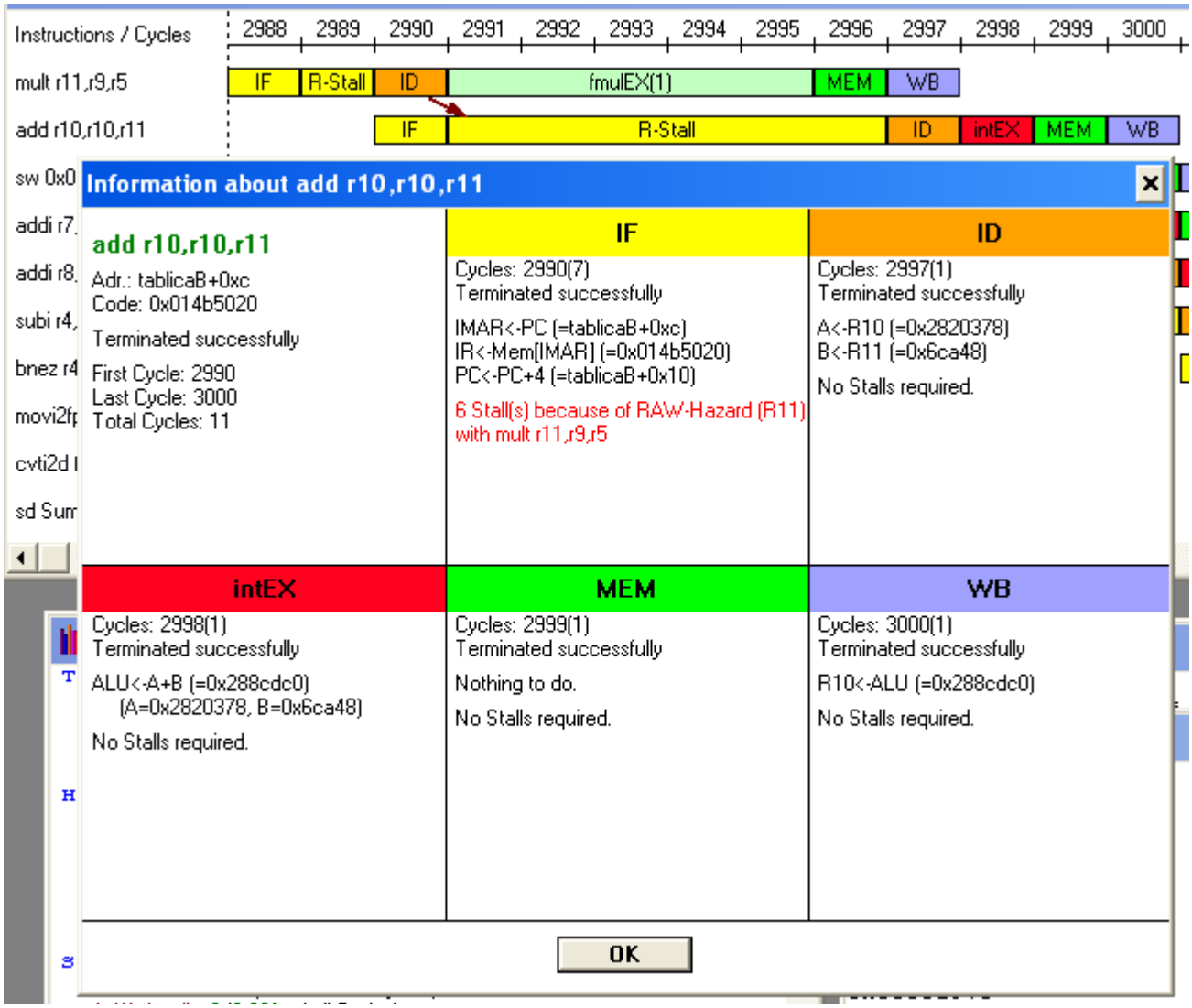


Czerwona strzałka wskazuje miejsce, w którym konieczne jest zastosowanie „R-Stall”. Przyczyną tego jest to, że instrukcja wskazywana przez strzałkę (bnez) musi poczekać dodatkową ilość cykli (2), żeby poprzednia instrukcja (subi) mogła się wykonać (zjawisko hazardu). Przyczyna leży po stronie instrukcji poprzedniej, ponieważ wymagała ona więcej czasu na działanie, stąd instrukcja „bnez” musi czekać. W przypadku zastosowania forwardingu oprogramowanie wykorzystałoby wynik poprzedniej instrukcji do obliczeń, co skróciłoby czas wykonania tej instrukcji (w tym przypadku o 1 cykl).

Poniżej przedstawiony jest diagram uwzględniający forwarding, obrazujący skrócenie czasu wykonania o 1 cykl (zaznaczone przy pomocy zielonej strzałki).

Zastosowana operacja:

instrukcja „add r10, r10, r11”



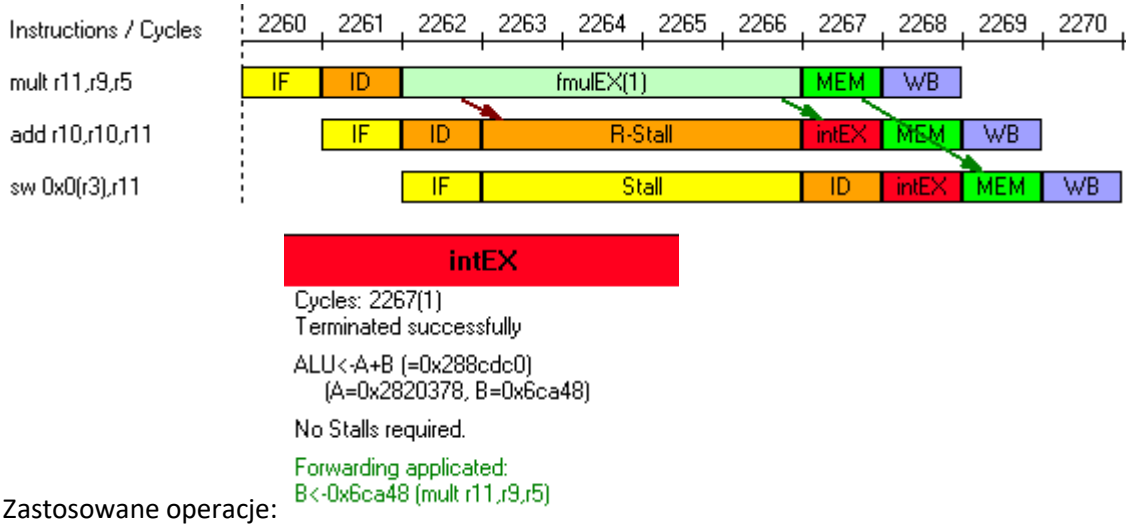
Ponieważ mnożenie jest dość złożone pod względem czasu wykonania, a instrukcja „add” potrzebuje tego wyniku do wykonania swojej instrukcji (potrzebuje wartości rejestru r11, do którego właśnie zwracany jest wynik mnożenia), musi ona się wstrzymać (dokładnie na 6 cykli). Dopiero po zakończeniu mnożenia, wznawia działanie.

W przypadku zastosowania forwardingu czas wstrzymania zostanie zmniejszony o 2 cykle, ponieważ instrukcja dodawanie wykorzysta wartości obliczone w trakcie wykonania instrukcji mnożenia. W międzyczasie rozpoczyna się działanie kolejnej instrukcji (sw), która również potrzebuje do wykonania wartości rejestru r4, w

związku z tym również zostaje wstrzymana. Nie dzieje się to w przypadku wyłączonego forwardingu, gdyż tam instrukcja ta rozpoczyna swoje działanie dopiero po wznowieniu instrukcji „add”.

Poniżej przedstawiony jest diagram uwzględniający forwarding, z widoczną różnicą ilości wymaganych cykli.

Zastosowane operacje:



G. Wnioski – wypływ forwardingu na działanie programu

1. Forwarding nie ma wpływu na wynik końcowy – wartość obliczonej sumy jest niezmienna
2. Przez zaistnienie zjawiska hazardu w danym momencie programu instrukcje muszą zostać wstrzymywane na jakiś czas (Stall), aby poprzednie były w stanie się w pełni wykonać.
3. Wstrzymanie wykonania instrukcji wynika z tego, że korzysta ona z rejestrów, które wykorzystywane są przez instrukcję poprzedzającą. Bez wstrzymania mogłoby dojść do nieścisłości, podczas których zostałaby pobrana błędna wartość.
4. W celu optymalizacji ilości cykli potrzebnych do wykonania programu można zastosować forwarding. Wykorzystuje on ponownie wyniki poprzedniej operacji, co skraca czas jego wykonania.
5. Niewielkie optymalizacje wykonane przez forwarding wewnątrz pętli instrukcji zaczynają mieć kardynalne znaczenie w ostatecznym czasie działania programu.