КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: КОМПЮТЪРНИ АРХИТЕКТУРИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 9

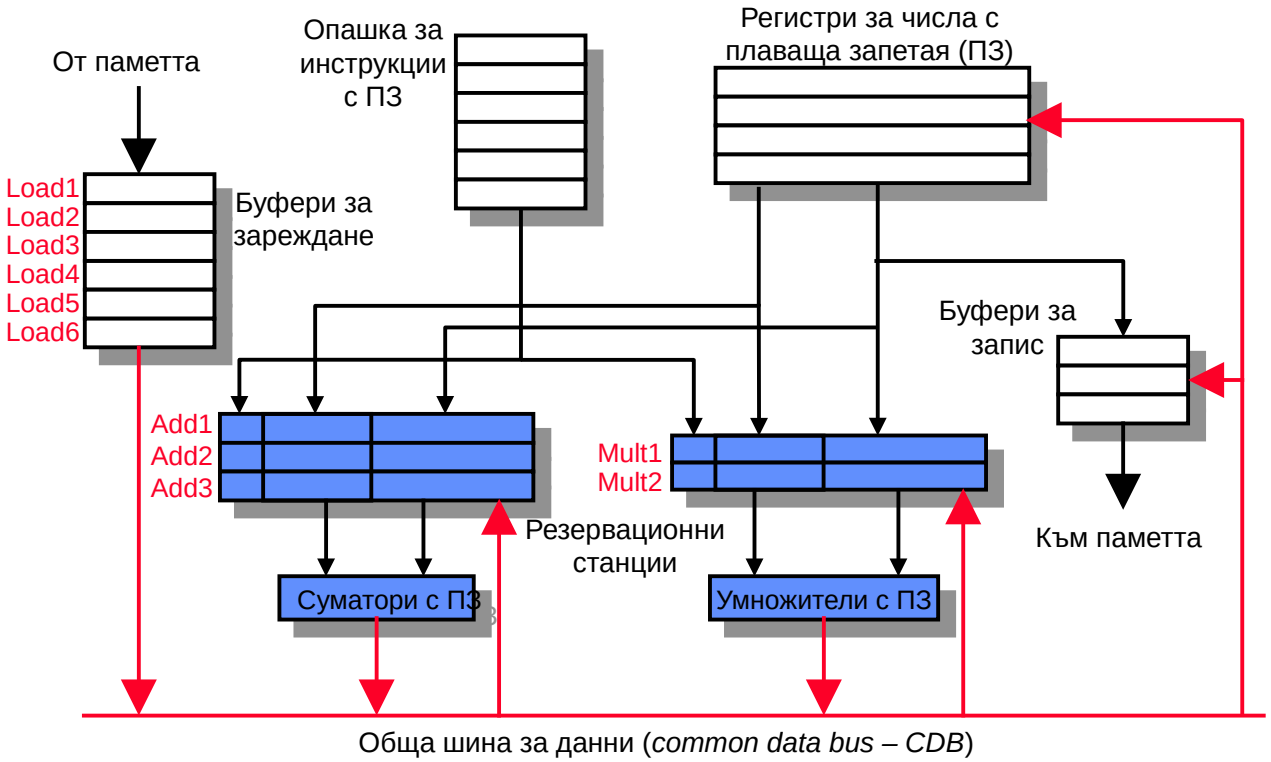
ТЕМА: КОНВЕЙЕРНО ИЗПЪЛНЕНИЕ С ПРЕНАРЕЖДАНЕ НА ИНСТРУКЦИИТЕ - АЛГОРИТЪМ НА ТОМАСУЛО

Цел: Да даде практически знания на студентите за работата на алгоритъма на Томасуло за динамично планиране (пренареждане) на инструкциите.

I. Теоретична част:

Алгоритъмът на Томасуло бе представен в предходната лекция. Тук ще повторим неговите особености и ще разгледаме един пример, който да поясни начина му на работа.

В сравнение с централизираното управление в „Scoreboard“, в алгоритъма на Томасуло се реализира разпределено управление на инструкциите. Първо, средствата за откриване на състезанията в потока инструкции са разпределени – те се намират в така наречените ***„резервационни станции“*** (Reservation Stations – RS). Станциите за резервиране се намират преди всяко функционално устройство. Чрез тях се определя кога може да стартира изпълнението на инструкцията върху функционалното устройство. Второ, резултатите от извършените операции се изпращат директно във функционалните устройства без да е необходимо да преминават през регистрите с общо предназначение. За целта се използва общата шина за данни (*common data bus – CDB*), по която резултатът достига едновременно до всички устройства, но се използва само от тези, които очакват този операнд. На фигурата по-долу е показана организацията на устройствата, реализиращи алгоритъма на Томасуло.



В станциите за резервиране се съхраняват инструкциите, които трябва да се изпълнят в съответните функционални устройства. Информацията, необходима за управление на започналото изпълнение на съответната инструкция в устройството, също се съхранява в резервационните станции. Буферите за зареждане и за запис съхраняват постъпилите от паметта данни и готовите за запис в паметта резултати. Регистрите за числа с плаваща запетая са свързани към резервационните станции с две шини за данни, а с буферите за зареждане и за запис – чрез общата шина за данни – CDB. Така всички резултати от функционалните устройства и от паметта се разпространяват чрез CDB-шината до входовете

на всички устройства с изключение на буфера за зареждане. Всички буфери и станции за резервиране съдържат полета за данни, използвани при управление на състезанията.

Изпълнението на една инструкция, в представената структура, преминава през следните три етапа:

1. *Допускане*. Взема се инструкция от опашката за инструкции с плаваща запетая. Ако инструкцията е за изчислителна операция (събиране, умножение и т. н.), тя се зарежда в свободна станция за резервиране, заедно с нейните операнди, ако те се съдържат в регистрите за числа с плаваща запетая. Ако инструкцията е за зареждане (*load*) или запис (*store*), тя може да се допусне, ако има свободен буфер. При липса на свободна станция за резервиране или на свободен буфер възниква структурен конфликт и инструкцията се задържа до тогава, докато не се освободи станция или буфер.

2. *Изпълнение*. Ако единият или и двата операнда на инструкцията по някакви причини не са налични, то се изчаква края на изчислението на стойността за необходимия регистър и появата ѝ на *CDB*-шината. Когато двата операнда са налични, то операцията на допуснатата инструкция се изпълнява. По този начин се избягват състезанията от типа *RAW*.

3. *Запис на резултата*. Когато изпълнението на операцията приключи, резултатът от нея се подава на *CDB*-шината, чрез която се прехвърля в регистър за числа с плаваща запетая или в онази станция за резервиране, която обслужва очакващото този резултат функционално устройство.

Въпреки че формулираните по-горе действия са сходни с тези при централизираното управление със „*Scoreboard*“, тук има три важни различия: 1) липсва предотвратяване на състезания от тип *WAW* и *WAR*, тъй като това се явява страничен ефект на алгоритъма; 2) за предаване на резултатите се използва общата шина (*CDB*-шината), а не схема за очакване на готовността на регистрите. 3) буферите за зареждане и за запис се разглеждат като основни функционални устройства.

Всяка станция за резервиране съдържа 6 полета с данни:

Op - операция, която следва да се изпълняваната върху операндите-източници S1 и S2;

$\boldsymbol{Q_j}$ **,** $\boldsymbol{Q_k}$ - станции за резервиране, които ще издават съответния операнд-източник. Нулата показва, че стойността на операнда-източник вече е достъпна в полето V_j или V_k , или е незадължителна.

$\boldsymbol{V_j}$ **,** $\boldsymbol{V_k}$ - стойността на операнда-източник. За даден операнд само едно от полетата V или Q е действително.

Busy - признак, който показва, че дадената станция и съответстващото ѝ функционално устройство, са заети.

За да се укаже кое функционално устройство в кой регистър от регистровия файл ще записва получения резултат се поддържа структура, наречена „*Register result status*“. Номерата на позициите в тази структура отговарят на номерата на регистрите в регистровия файл. Ако към определен момент дадена позиция е празна (т.е. има стойност *0*), то това означава, че активната инструкция не изчислява резултат, предназначен за този регистър.

Във всеки от буферите за зареждане и за запис съществува поле, съдържащо признак (флаг) за заетост. Този флаг показва кога съответният буфер става достъпен, благодарение на завършило зареждане или запис, назначено върху него.

II. Практическа част:

Ще разгледаме същата последователност от инструкции, която беше представена в предходното упражнение.

```
1:    LD    F6, 34(R2)    # F6  ←  MEM[R2+34]
2:    LD    F2, 45(R3)    # F2  ←  MEM[R3+45]
3:    MULTI F0, F2, F4    # F0  ←  F2 * F4
4:    SUBD  F8, F6, F2    # F8  ←  F6 - F2
5:    DIVD  F10,F0, F6    # F10 ←  F0 / F6
6:    ADD   F6, F8, F2    # F6  ←  F8 + F2
```

```
Анти-зависимост:
      - Зависимости по име:  4-6; 5-6;
      - Зависимости по изход: 1-6;
Зависимости по данни: 1-4; 1-5; 2-3; 2-4; 2-6; 3-5; 4-6
```

Броят на процесорните цикли, необходими за изпълнението на операциите са: зареждане от паметта (*LD*) – 1 цикъл; събиране и изваждане (*ADD/SUB*) – 2 цикъла; умножение (*MULT*) – 10 цикъла; деление (*DIV*) – 40 цикъла.

Първоначално структурите от данни, съдържащи информация за управление на изпълнението са празни.

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2			
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status:

Cycle		F0	F2	F4	F6	F8	F10	F12	...	F30
0	FU									

След първия и след втория процесорен цикъл структурите от данни в алгоритъма Томасуло, изглеждат така:

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34 + R2
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status:

Cycle		F0	F2	F4	F6	F8	F10	F12	...	F30
1	FU	Load1								

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34 + R2
Load2	Yes	45 + R3
Load3	No	

За разлика от „Scoeboard“ тук няколко инструкции Load се допускат едновременно до планиране.

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	Load2	
	Mult2	No					

Register result status:

Cycle		F0	F2	F4	F6	F8	F10	F12	...	F30
3	FU	Mult1	Load2		Load1					

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45 + R3
Load3	No	

Инструкцията MULT се зарежда в резервационната станция Mult1. Тъй като операндът F4 е готов, неговата стойност, обозначена с R(F4) също се зарежда. За другия операнд – F2 следва да се изчака, докато стойността му се появи на CDB. Това ще се случи след като Load2 приключи изпълнението си. Load1 приключва изпълнението си.

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	Yes	SUBD	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	Load2	
	Mult2	No					

Register result status:

Cycle		F0	F2	F4	F6	F8	F10	F12	...	F30
4	FU	Mult1	Load2		M(A1)	Add1				

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Инструкцията Load1 е приключила изпълнението си. Стойността на клетката от паметта, обозначена чрез M(A1) и заредена чрез Load1 се е появила на CDB и се записва в регистъра F6.
Инструкцията SUB се зарежда в станцията Add1, като се зарежда и първият ѝ операнд-източник – записаната стойност в F6.
Load2 приключва изпълнението си.

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
2	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Cycle		F0	F2	F4	F6	F8	F10	F12	...	F30
5	FU	Mult1	M(A2)		M(A1)	Add1	Mult2			

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1		
LD	F2	45+	R3	2		
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34 + R2
Load2	Yes	45 + R3
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Инструкцията Load2 е приключила изпълнението си. Стойността на клетката от паметта, обозначена чрез M(A2) и заредена чрез Load2 се е появила на CDB и се записва в регистъра F2, както и в двете очакващи я резервационни станции - Add1 и Mult1. Инструкцията DIV се зарежда в станцията Mult2, като се зарежда и вторият ѝ операнд-източник – готовата стойност от F6.

Register result status:

Cycle		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
2	*FU*		Load2		Load1					

В петия процесорен цикъл стойностите на операндите-източници за двете инструкции *SUB* и *MULT* стават налични. Това означава, че може да започне изпълнението на двете операции от съответните функционални устройства. Времената за приключване на двете операции са: *SUB* – 2 процесорни цикъла; *MULT* – 10 процесорни цикъла.

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
1	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Продължава изпълнението на инструкциите SUB и MULT.
Инструкцията ADD се зарежда в станцията Add2, като се зарежда и вторият ѝ операнд-източник – записаната стойност в F2.
Другият ѝ операнд-източник се очаква по CDB, след приключването на инструкцията SUB.

Register result status:

Cycle		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
6	*FU*	Mult1	M(A2)		Add2	Add1	Mult2			

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Приключва изпълнението на инструкцията SUB.

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Cycle		F0	F2	F4	F6	F8	F10	F12	...	F30
7	FU	Mult1	M(A2)		Add2	Add1	Mult2			

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Резултатът от изпълнението на инструкцията SUB е наличен на CDB. Стойността, означена с (M-M) се записва в регистъра F8 и в резервационната станция Add2. Започва изпълнението на ADD, тъй като стойностите на двата операнда-източници вече са налични. Изпълнението на инструкцията MULT продължава.

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
2	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
7	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Cycle		F0	F2	F4	F6	F8	F10	F12	...	F30
8	FU	Mult1	M(A2)		Add2	(M-M)	Mult2			

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Инструкциите ADD и MULT се изпълняват.

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
6	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Cycle		F0	F2	F4	F6	F8	F10	F12	...	F30
9	FU	Mult1	M(A2)		Add2	(M-M)	Mult2			

Приключва изпълнението на инструкцията ADD.

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Cycle		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
10	*FU*	Mult1	M(A2)		Add2	(M-M)	Mult2			

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Cycle		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
11	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Резултатът от изпълнението на инструкцията ADD е наличен на CDB. Стойността, означена с (M-M+M) се записва в регистъра F6. Инструкцията MULT ще приключи изпълнението си след 4 цикъла. Инструкцията DIV не може да започне да се изпълнява, тъй като първият ѝ операнд-източник е зависим от резултата от инструкцията MULT.

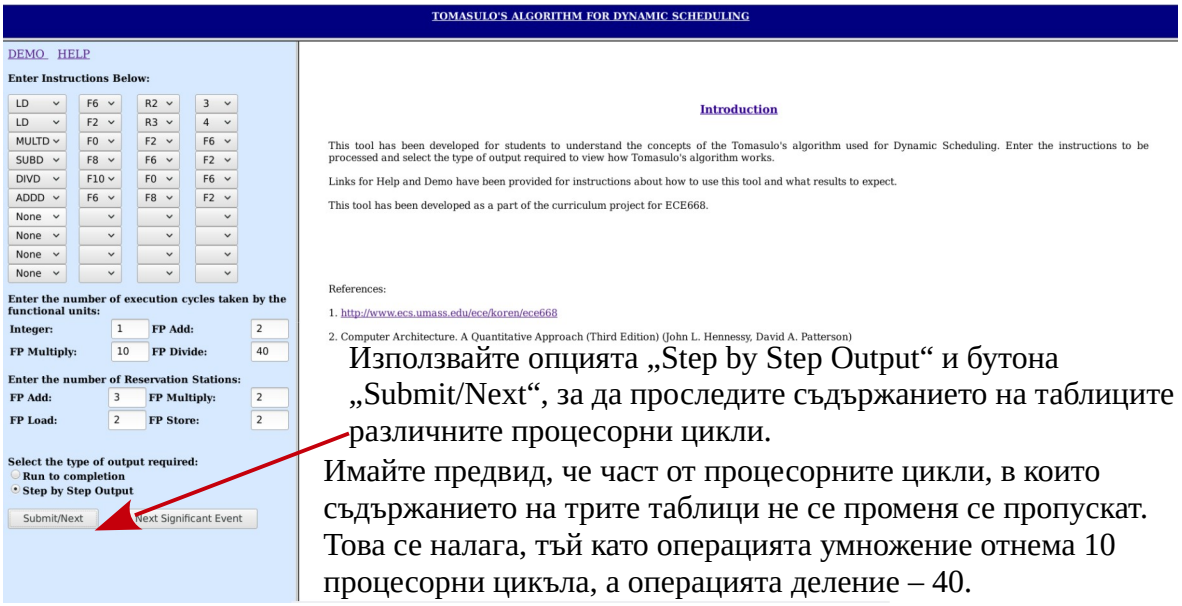
На този етап приключва изпълнението на инструкциите с продължителност до два процесорни цикъла. Продължава изпълнението на продължителните операции умножение и деление. В петнадесетия процесорен цикъл инструкцията *MULT* ще приключи изпълнението си. В шестнадесетия процесорен цикъл резултатът от операцията ще се бъде наличен на *CDB*-шината и стойността ще се прехвърли в регистъра *F0* и в резервационната станция Mult2, което ще даде възможност за започне изпълнението на инструкцията *DIV*. В процесорен цикъл номер 56 ще приключи изпълнението и на операцията деление на числа с плаваща запетая. На цикъл 57 резултатът от тази операция ще се предаде по *CDB*-шината и ще се запише в целевия регистър за тази инструкция – *F10*.

III. Задачи за изпълнение:

Задача 1:

Проследете как завършва изпълнението на програмата, представена в практическата част на това упражнение, като посетите следния адрес:

https://web.archive.org/web/20120622164200/http://www.ecs.umass.edu/ece/koren/architecture/Tomasulo1/tomasulo_files/tomasulo.htm



Използвайте опцията „Step by Step Output“ и бутона „Submit/Next“, за да проследите съдържанието на таблиците в различните процесорни цикли.

Имайте предвид, че част от процесорните цикли, в които съдържанието на трите таблици не се променя се пропускат. Това се налага, тъй като операцията умножение отнема 10 процесорни цикъла, а операцията деление – 40.

Отговорете на слените въпроси:

- След колко процесорни цикъла може да започне повторно изпълнение на програмата? (т.е. кога първата инструкция *LD* може да бъде допусната до планиране, отново за втори път)
- Как влияе пренареждането на времето за изпълнение на тази програма?
- Възникват ли състезания от вида Write-after-Read (WAR)? Как се преодоляват?

Задача 2:

Заредете следния адрес във Вашия браузер:
https://web.archive.org/web/20120622164200/http://www.ecs.umass.edu/ece/koren/architecture/Tomasulo1/tomasulo_files/tomasulo.htm

Въведете кода на следващата програма от 6 инструкции. Изпълнете я постъпково и проследете състоянието на инструкциите в трите таблици.

```
LD    F0, 0(R1)
ADDD  F4, F0, F2
SD    F4, 0(R1)
LD    F0, 4(R1)
ADDD  F4, F0, F2
SD    F4, 4(R1)
```

Отговорете на следните въпроси:

- След колко процесорни цикъла тази програма може да започне да се изпълнява отначало?
- Как влияе пренареждането на времето за изпълнение на тази програма?
- Възникват ли състезания от вида Write-after-Read (WAR)? Как се преодоляват?
- Запишете зависимостите между инструкциите в програмата, като следвате примера от практическата част на това упражнение.

Задача 3: Въведете кода и проследете изпълнението на следващата програма. В нея се сумират произведения. Използва се за реализация на бързо преобразование на Фурие (Fast Fourier Transform).

```
LD    F0, 0(R1)
LD    F2, 2(R1)
MULTD F8, F0, F2
LD    F4, 4(R1)
LD    F6, 6(R1)
MULTD F10, F4, F6
ADDD  F10, F8, F10
LD    F8, 0(R1)
ADDD  F10, F10, F8
```

Отговорете на следните въпроси:

- След колко процесорни цикъла тази програма може да започне да се изпълнява отначало?
- Как влияе пренареждането на времето за изпълнение на тази програма?
- Възникват ли състезания от вида Write-after-Read (WAR)? Как се преодоляват?
- Запишете зависимостите между инструкциите в програмата, като следвате примера от практическата част на това упражнение.