Ayden Dauenhauer

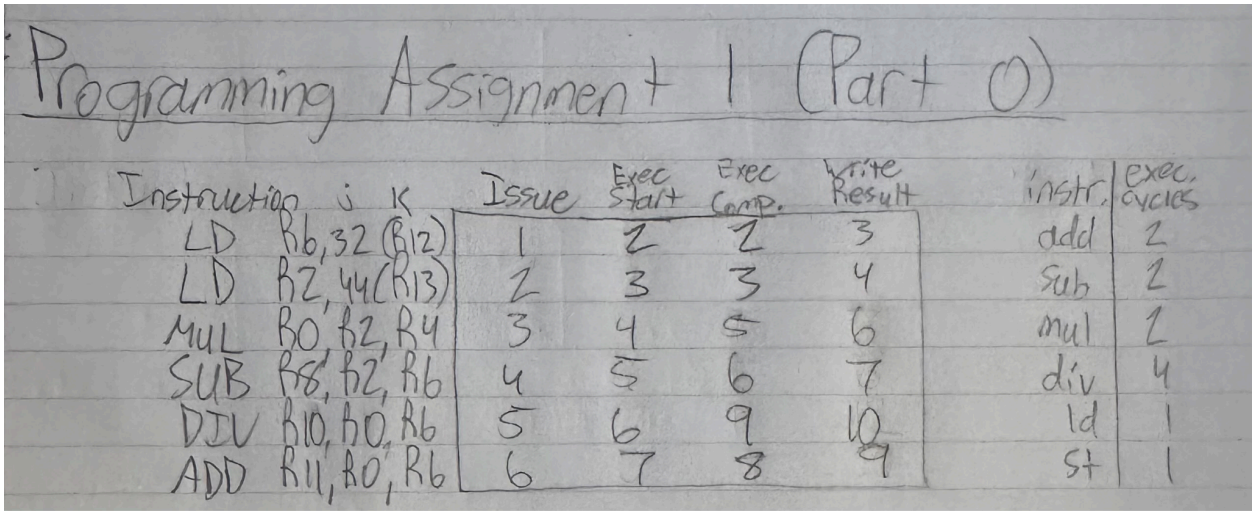
Prof. Grover

ECEN 511

17 November 2025

Programming Assignment 1

PART 0: Understand the Given Architecture



Programming Assignment 1 (Part 0)

Instruction j k	Issue	Exec Start	Exec Comp.	Write Result
LD R6,32(R12)	1	2	2	3
LD R2,44(R13)	2	3	3	4
MUL R0,R2,R4	3	4	5	6
SUB R8,R2,R6	4	5	6	7
DIV R10,R0,R6	5	6	9	10
ADD R11,R0,R6	6	7	8	9

instr.	exec. cycles
add	2
sub	2
mul	2
div	4
ld	1
st	1

Fig 1: Cycles to Complete Execution

* It is expected that the instructions will take 10 cycles to complete

PART 1: Implement the Basic Algorithm

```
/*******************************
*     Step III: Write result
*******************************/
for(i=0;i<NUM_RS_ENTRIES;i++) {
 // complete STEP III here
 RS * curr_rs = get_rs(i+1);
 if(curr_rs->is_result_ready == true) {
   if(curr_rs->op==ADD || curr_rs->op==SUB || curr_rs->op==MUL ||
curr_rs->op==DIV || curr_rs->op==LD) {
     for(j=0;j<NUM_REGS;j++) {
       if(regs[j].Qi == curr_rs->id) {
         regs[j].val = curr_rs->result;
         regs[j].Qi = 0;
       }
     }
     for(k=0;k<NUM_RS_ENTRIES;k++) {
       RS * dependent_rs = get_rs(k+1);
       if(dependent_rs->Qj == curr_rs->id) {
         dependent_rs->Vj = curr_rs->result;
         dependent_rs->Qj = 0;
       }
       if(dependent_rs->Qk == curr_rs->id) {
         dependent_rs->Vk = curr_rs->result;
         dependent_rs->Qk = 0;
       }
     }
   }
   else if(curr_rs->op==ST) {
     if(curr_rs->Qk == 0) {
       set_mem(curr_rs->Vj + curr_rs->A, curr_rs->Vk);
     }
   }
   reset_rs_entry(curr_rs);
 }
}
```

* This write stage writes the result of the exec stage once the exec stage marks that the result has been calculated and is ready. Also it updates the rest of the reservation station to let the rest of the instructions know that the register was updated in case there are dependent registers.

```
/*******************************
*     Step II: Execute
*******************************/
for(i=0;i<NUM_RS_ENTRIES;i++) {
 // complete STEP II here
 RS * curr_rs = get_rs(i+1);
 if(curr_rs->is_busy == true){
   if(curr_rs->op==ADD || curr_rs->op==SUB || curr_rs->op==MUL ||
curr_rs->op==DIV) {
     if(curr_rs->Qj == 0 && curr_rs->Qk == 0) {
       curr_rs->in_exec = true;
       if(curr_rs->exec_cycles==1) {
         if(curr_rs->op==ADD) {
           curr_rs->result = curr_rs->Vj + curr_rs->Vk;
         }
         else if(curr_rs->op==SUB) {
           curr_rs->result = curr_rs->Vj - curr_rs->Vk;
         }
         else if(curr_rs->op==MUL) {
           curr_rs->result = curr_rs->Vj * curr_rs->Vk;
         }
         else if(curr_rs->op==DIV) {
           curr_rs->result = curr_rs->Vj / curr_rs->Vk;
         }
         curr_rs->is_result_ready = true;
       }
       else {
         curr_rs->exec_cycles = curr_rs->exec_cycles - 1;
       }
     }
   }
   else if(curr_rs->op==LD || curr_rs->op==ST) {
     if(curr_rs->Qj == 0) {
       curr_rs->in_exec = true;
       if(curr_rs->exec_cycles==1) {
         if(curr_rs->op==LD) {
           curr_rs->result = get_mem(curr_rs->Vj + curr_rs->A);
         }
         if(curr_rs->op==ST) {
           //set_mem(curr_rs->Vj + curr_rs->A, curr_rs->Vk);
         }
         curr_rs->is_result_ready = true;
       }
       else {
         curr_rs->exec_cycles = curr_rs->exec_cycles - 1;
       }
     }
   }
 }
```

```
}
```

* This exec stage computes the result of the operations at the proper execution cycle. Also it does not begin unless the dependent registers (Qj and Qk for ADD, SUB, MUL, DIV. Just Qj for LD and ST).

Default Instructions and Execution Cycle Values

```
============== TEST INSTRUCTION SEQUENCE ===========
I#1     ld      r6,32(r12)
I#2     ld      r2,44(r13)
I#3     mul     r0,r2,r4
I#4     sub     r8,r2,r6
I#5     div     r10,r0,r6
I#6     add     r11,r0,r6
* CYCLE 0 (initial state)
=================================================================================================
RS_id   type    Busy    inst#   Op      Vj      Vk      Qj      Qj      A       Exec    Done
-------------------------------------------------------------------------------------------------
#1      LD      No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#2      LD      No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#3      ST      No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#4      ST      No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#5      ADD     No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#6      ADD     No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#7      ADD     No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#8      MUL     No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#9      MUL     No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
=================================================================================================
================================================================
Registers
----------------------------------------------------------------
        r0      r1      r2      r3      r4      r5      r6      r7
----------------------------------------------------------------
val     0       1       2       3       4       5       6       7
Qi      #0      #0      #0      #0      #0      #0      #0      #0
----------------------------------------------------------------
        r8      r9      r10     r11     r12     r13     r14     r15
----------------------------------------------------------------
val     8       9       10      11      12      13      14      15
Qi      #0      #0      #0      #0      #0      #0      #0      #0
----------------------------------------------------------------
================================================================
* CYCLE 1
=================================================================================================
RS_id   type    Busy    inst#   Op      Vj      Vk      Qj      Qj      A       Exec    Done
-------------------------------------------------------------------------------------------------
#1      LD      Yes     I#-1    ld      12      -1      #0      #0      32      No      No
#2      LD      No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
#3      ST      No      I#-1    NONE    -1      -1      #0      #0      -1      No      No
```

#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7
Qi	#0	#0	#0	#0	#0	#0	#1	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 2

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	Yes	I#-1	ld	12	-1	#0	#0	32	Yes	Yes
#2	LD	Yes	I#-1	ld	13	-1	#0	#0	44	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7
Qi	#0	#0	#2	#0	#0	#0	#1	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 3

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	Yes	I#-1	ld	13	-1	#0	#0	44	Yes	Yes
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	-1	4	#2	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	12	7
Qi	#8	#0	#2	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 4

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	sub	9	12	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	9	4	#0	#0	-1	Yes	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	9	3	4	5	12	7
Qi	#8	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#5	#0	#0	#0	#0	#0	#0	#0

* CYCLE 5

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	sub	9	12	#0	#0	-1	Yes	No

#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	9	4	#0	#0	-1	Yes	Yes
#9	MUL	Yes	I#-1	div	-1	12	#8	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	9	3	4	5	12	7
Qi	#8	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#5	#0	#9	#0	#0	#0	#0	#0

* CYCLE 6

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	sub	9	12	#0	#0	-1	Yes	Yes
#6	ADD	Yes	I#-1	add	36	12	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	36	12	#0	#0	-1	Yes	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#5	#0	#9	#6	#0	#0	#0	#0

* CYCLE 7

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	Yes	I#-1	add	36	12	#0	#0	-1	Yes	No

#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	36	12	#0	#0	-1	Yes	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	-3	9	10	11	12	13	14	15
Qi	#0	#0	#9	#6	#0	#0	#0	#0

* CYCLE 8

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	Yes	I#-1	add	36	12	#0	#0	-1	Yes	Yes
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	36	12	#0	#0	-1	Yes	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	-3	9	10	11	12	13	14	15
Qi	#0	#0	#9	#6	#0	#0	#0	#0

* CYCLE 9

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	36	12	#0	#0	-1	Yes	Yes

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	-3	9	10	48	12	13	14	15
Qi	#0	#0	#9	#0	#0	#0	#0	#0

* CYCLE 10

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	-3	9	3	48	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* The results are expected since it matches the expected cycle results from Fig 1.

Looking instruction by instruction: The first LD is issued at cycle 1, starts and completes execution at cycle 2, and writes back at cycle 3. The second LD is issued at cycle 2, starts and completes execution at cycle 3, and writes back at cycle 4. MUL is issued at cycle 3, starts

execution at cycle 4 and finishes at cycle 5, and writes back at cycle 6. SUB is issued at cycle 4, starts execution at cycle 5 and finishes at cycle 6, and writes back at cycle 7. DIV is issued at cycle 5, starts execution at cycle 6 and finishes at cycle 9, and writes back at cycle 10. ADD is issued at cycle 6, starts execution at cycle 7 and finishes at cycle 8, and writes back at cycle 9. Thus we have confirmed that the instructions will take 10 cycles to complete.

Also, the proper values get stored in their respective destination registers. R0 = 36, R2 = 9, R6 = 12, R8 = -3, R10 = 3, and R11 = 48.

Default Instructions with LD_LAT = 2

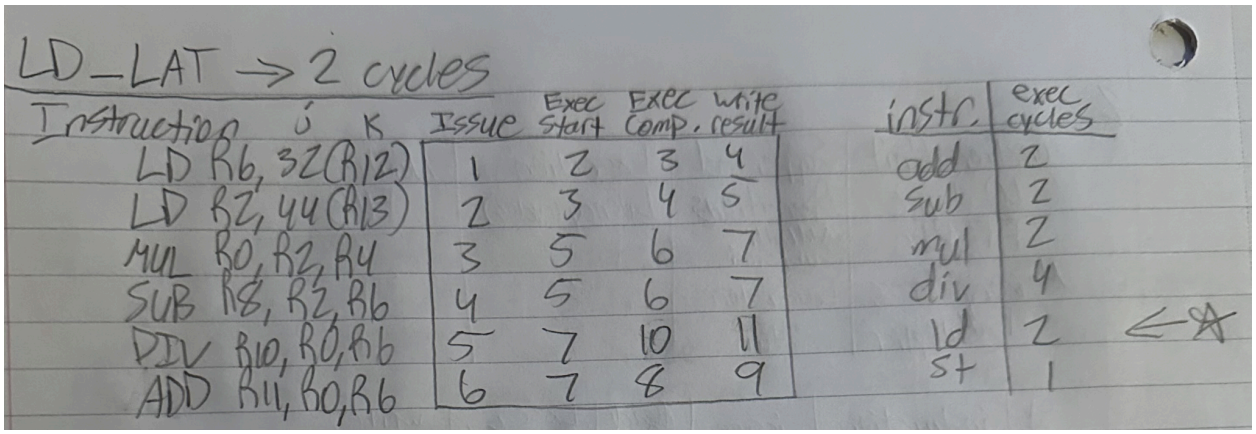


Fig 2: Cycles to Complete Execution with LD_LAT = 2

* It is expected that the instructions will take 11 cycles to complete

```
/* execution unit latencies */
#define LAT_ADD 2 /* executed on ADD */
#define LAT_SUB 2 /* executed on ADD */
#define LAT_MUL 2 /* executed on MUL */
#define LAT_DIV 4 /* executed on MUL */
#define LAT_LD 2 /* executed on Memory Unit */
#define LAT_ST 1 /* executed on Memory Unit */
```

* LD_LAT is updated to 2 in the arch.h file

========== TEST INSTRUCTION SEQUENCE ==========

```
I#1     ld      r6,32(r12)
I#2     ld      r2,44(r13)
I#3     mul     r0,r2,r4
I#4     sub     r8,r2,r6
I#5     div     r10,r0,r6
I#6     add     r11,r0,r6
```

* CYCLE 0 (initial state)

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 1

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	Yes	I#-1	ld	12	-1	#0	#0	32	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7

Qi	#0	#0	#0	#0	#0	#0	#1	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 2

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	Yes	I#-1	ld	12	-1	#0	#0	32	Yes	No
#2	LD	Yes	I#-1	ld	13	-1	#0	#0	44	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7
Qi	#0	#0	#2	#0	#0	#0	#1	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 3

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	Yes	I#-1	ld	12	-1	#0	#0	32	Yes	Yes
#2	LD	Yes	I#-1	ld	13	-1	#0	#0	44	Yes	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	-1	4	#2	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7
Qi	#8	#0	#2	#0	#0	#0	#1	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 4

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	Yes	I#-1	ld	13	-1	#0	#0	44	Yes	Yes
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	sub	-1	12	#2	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	-1	4	#2	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	12	7
Qi	#8	#0	#2	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#5	#0	#0	#0	#0	#0	#0	#0

* CYCLE 5

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	sub	9	12	#0	#0	-1	Yes	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	9	4	#0	#0	-1	Yes	No
#9	MUL	Yes	I#-1	div	-1	12	#8	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	9	3	4	5	12	7
Qi	#8	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#5	#0	#9	#0	#0	#0	#0	#0

* CYCLE 6

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	sub	9	12	#0	#0	-1	Yes	Yes
#6	ADD	Yes	I#-1	add	-1	12	#8	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	9	4	#0	#0	-1	Yes	Yes
#9	MUL	Yes	I#-1	div	-1	12	#8	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	9	3	4	5	12	7
Qi	#8	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#5	#0	#9	#6	#0	#0	#0	#0

* CYCLE 7

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	Yes	I#-1	add	36	12	#0	#0	-1	Yes	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	36	12	#0	#0	-1	Yes	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15

val	-3	9	10	11	12	13	14	15
Qi	#0	#0	#9	#6	#0	#0	#0	#0

* CYCLE 8

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	Yes	I#-1	add	36	12	#0	#0	-1	Yes	Yes
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	36	12	#0	#0	-1	Yes	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	-3	9	10	11	12	13	14	15
Qi	#0	#0	#9	#6	#0	#0	#0	#0

* CYCLE 9

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	36	12	#0	#0	-1	Yes	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15

val	-3	9	10	48	12	13	14	15
Qi	#0	#0	#9	#0	#0	#0	#0	#0

* CYCLE 10

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	36	12	#0	#0	-1	Yes	Yes

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	-3	9	10	48	12	13	14	15
Qi	#0	#0	#9	#0	#0	#0	#0	#0

* CYCLE 11

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	36	1	9	3	4	5	12	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	-3	9	3	48	12	13	14	15

```
Qi          #0          #0          #0          #0          #0          #0          #0          #0
-------------------------------------------------------------------------
=========================================================================
```

* The results are expected since it matches the expected cycle results from Fig 2.

Looking instruction by instruction: The first LD is issued at cycle 1, starts execution at cycle 2, and now completes at cycle 3, and writes back at cycle 4. The second LD is issued at cycle 2, starts execution at cycle 3, and now completes at cycle 4, and writes back at cycle 5. MUL is issued at cycle 3, and now starts execution at cycle 5 since it has to wait for the second LD to finish since R2 is dependent. It now finishes at cycle 6, and writes back at cycle 7. SUB doesn't change. DIV is issued at cycle 5, and now starts execution at cycle 7 since it has to wait for MUL to finish since R0 is dependent. It now finishes at cycle 10, and writes back at cycle 11. ADD doesn't change.

Also, the proper values get stored in their respective destination registers. R0 = 36, R2 = 9, R6 = 12, R8 = -3, R10 = 3, and R11 = 48. Just like the first LD_LAT scenario.

All Six Instruction Types With At Least Three Data Dependencies

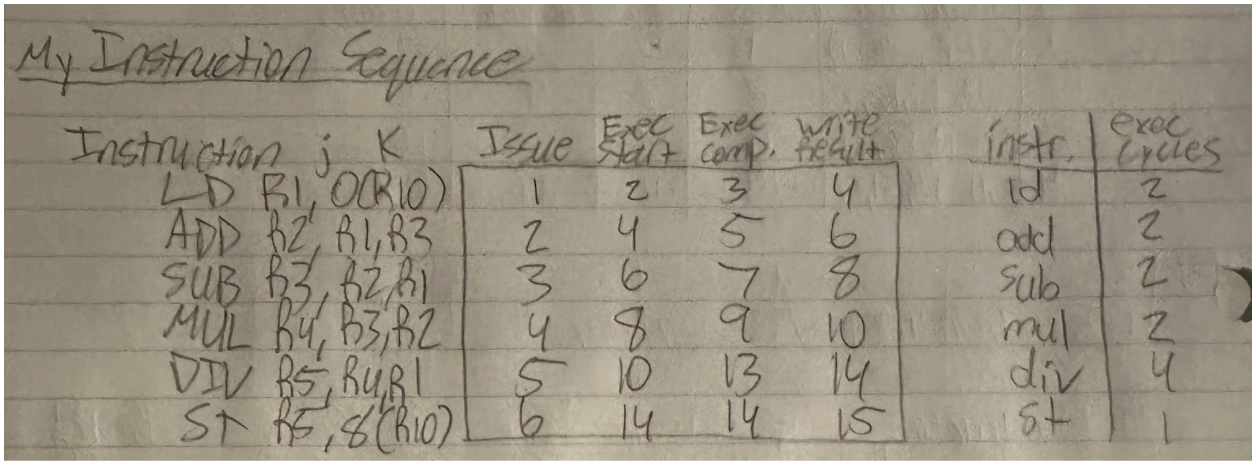


Fig 3: My Instruction Sequence with All Six Instruction Types and At Least Three Data Dependencies

* It is expected that the instructions will take 15 cycles to complete

```
INST inst[NUM_OF_INST]; /* instruction array */
/*****************************
* instruction initialization
*****************************/
void init_inst()
{
 /*inst[0].num=1; inst[0].op=LD; inst[0].rd=6; inst[0].rs=12; inst[0].rt=32; //
ld r6,32(r12)
 inst[1].num=2; inst[1].op=LD; inst[1].rd=2; inst[1].rs=13; inst[1].rt=44; //
ld r2,44(r13)
 inst[2].num=3; inst[2].op=MUL; inst[2].rd=0; inst[2].rs=2; inst[2].rt=4; //
mul r0, r2, r4
 inst[3].num=4; inst[3].op=SUB; inst[3].rd=8; inst[3].rs=2; inst[3].rt=6; //
sub r8, r2, r6
 inst[4].num=5; inst[4].op=DIV; inst[4].rd=10; inst[4].rs=0; inst[4].rt=6; //
div r10, r0, r6
 inst[5].num=6; inst[5].op=ADD; inst[5].rd=11; inst[5].rs=0; inst[5].rt=6; //
add r11, r0, r6 */
 inst[0].num=1; inst[0].op=LD; inst[0].rd=1; inst[0].rs=10; inst[0].rt=0; // ld
r1,0(r10)
 inst[1].num=2; inst[1].op=ADD; inst[1].rd=2; inst[1].rs=1; inst[1].rt=3; //
add r2, r1, r3
 inst[2].num=3; inst[2].op=SUB; inst[2].rd=3; inst[2].rs=2; inst[2].rt=1; //
sub r3, r2, r1
 inst[3].num=4; inst[3].op=MUL; inst[3].rd=4; inst[3].rs=3; inst[3].rt=2; //
mul r4, r3, r2
 inst[4].num=5; inst[4].op=DIV; inst[4].rd=5; inst[4].rs=4; inst[4].rt=1; //
div r5, r4, r1
 inst[5].num=6; inst[5].op=ST; inst[5].rd=10; inst[5].rs=5; inst[5].rt=8; // st
r5,8(r10)

 return;
}
```

* Instruction array is updated to match my instruction sequence in the inst.c file

```
============ TEST INSTRUCTION SEQUENCE ==========
I#1     ld      r1,0(r10)
I#2     add     r2,r1,r3
I#3     sub     r3,r2,r1
```

I#4 mul r4,r3,r2
I#5 div r5,r4,r1
I#6 st r10,8(r5)
* CYCLE 0 (initial state)

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 1

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	Yes	I#-1	ld	10	-1	#0	#0	0	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7
Qi	#0	#1	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 2

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	Yes	I#-1	ld	10	-1	#0	#0	0	Yes	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	add	-1	3	#1	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	1	2	3	4	5	6	7
Qi	#0	#1	#5	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 3

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	Yes	I#-1	ld	10	-1	#0	#0	0	Yes	Yes
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	add	-1	3	#1	#0	-1	No	No
#6	ADD	Yes	I#-1	sub	-1	-1	#5	#1	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7

val	0	1	2	3	4	5	6	7
Qi	#0	#1	#5	#6	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 4

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	add	10	3	#0	#0	-1	Yes	No
#6	ADD	Yes	I#-1	sub	-1	10	#5	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	-1	-1	#6	#5	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	2	3	4	5	6	7
Qi	#0	#0	#5	#6	#8	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 5

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	Yes	I#-1	add	10	3	#0	#0	-1	Yes	Yes
#6	ADD	Yes	I#-1	sub	-1	10	#5	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	-1	-1	#6	#5	-1	No	No
#9	MUL	Yes	I#-1	div	-1	10	#8	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	2	3	4	5	6	7
Qi	#0	#0	#5	#6	#8	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 6

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	-1	10	#9	#0	8	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	Yes	I#-1	sub	13	10	#0	#0	-1	Yes	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	-1	13	#6	#0	-1	No	No
#9	MUL	Yes	I#-1	div	-1	10	#8	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	4	5	6	7
Qi	#0	#0	#0	#6	#8	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 7

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	-1	10	#9	#0	8	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	Yes	I#-1	sub	13	10	#0	#0	-1	Yes	Yes
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	-1	13	#6	#0	-1	No	No
#9	MUL	Yes	I#-1	div	-1	10	#8	#0	-1	No	No

==========
=
==========

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	4	5	6	7
Qi	#0	#0	#0	#6	#8	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

==========

* CYCLE 8

==========
=

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	-1	10	#9	#0	8	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	3	13	#0	#0	-1	Yes	No
#9	MUL	Yes	I#-1	div	-1	10	#8	#0	-1	No	No

==========
=
==========

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	4	5	6	7
Qi	#0	#0	#0	#0	#8	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

==========

* CYCLE 9

==========
=

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	-1	10	#9	#0	8	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	Yes	I#-1	mul	3	13	#0	#0	-1	Yes	Yes
#9	MUL	Yes	I#-1	div	-1	10	#8	#0	-1	No	No

==========
=
==========

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	4	5	6	7
Qi	#0	#0	#0	#0	#8	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

==========

* CYCLE 10

==========
=

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	-1	10	#9	#0	8	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	39	10	#0	#0	-1	Yes	No

==========
=
==========

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	39	5	6	7
Qi	#0	#0	#0	#0	#0	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

==========

* CYCLE 11

==========
=

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	-1	10	#9	#0	8	No	No

#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	39	10	#0	#0	-1	Yes	No

==========
=
==========

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	39	5	6	7
Qi	#0	#0	#0	#0	#0	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

==========

* CYCLE 12

==========
=

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	-1	10	#9	#0	8	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	39	10	#0	#0	-1	Yes	No

==========
=
==========

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	39	5	6	7
Qi	#0	#0	#0	#0	#0	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

==========

* CYCLE 13

==========
=

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done

#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	-1	10	#9	#0	8	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	Yes	I#-1	div	39	10	#0	#0	-1	Yes	Yes

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	39	5	6	7
Qi	#0	#0	#0	#0	#0	#9	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 14

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	Yes	I#-1	st	3	10	#0	#0	8	Yes	Yes
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	39	3	6	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* CYCLE 15

RS_id	type	Busy	inst#	Op	Vj	Vk	Qj	Qj	A	Exec	Done
#1	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#2	LD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#3	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#4	ST	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#5	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#6	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#7	ADD	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#8	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No
#9	MUL	No	I#-1	NONE	-1	-1	#0	#0	-1	No	No

Registers

	r0	r1	r2	r3	r4	r5	r6	r7
val	0	10	13	3	39	3	6	7
Qi	#0	#0	#0	#0	#0	#0	#0	#0

	r8	r9	r10	r11	r12	r13	r14	r15
val	8	9	10	11	12	13	14	15
Qi	#0	#0	#0	#0	#0	#0	#0	#0

* The results are expected since it matches the expected cycle results from Fig 3.

Looking instruction by instruction: LD is issued at cycle 1, starts execution at cycle 2, completes at cycle 3, and writes back at cycle 4. ADD is issued at cycle 2, waits to start execution at cycle 4, completes at cycle 5, and writes back at cycle 6. SUB is issued at cycle 3, waits to start execution at cycle 6, completes at cycle 7, and writes back at cycle 8. MUL is issued at cycle 4, waits to start execution at cycle 8, completes at cycle 9, and writes back at cycle 10. DIV is issued at cycle 5, waits to start execution at cycle 10, completes at cycle 13, and writes back at cycle 14. ST is issued at cycle 6, waits to start execution at cycle 14, completes at cycle 14, and writes back at cycle 15.

Also, the proper values get stored in their respective destination registers. R1 = 10, R2 = 13, R4 = 39, R5 = 3. ST does work, however it doesn't display since the display only shows the registers and not the memory array.