LECTURE 10

Pipelining: Advanced ILP

EXCEPTIONS

- An *exception*, or *interrupt*, is an event other than regular transfers of control (branches, jumps, calls, returns) that changes the normal flow of instruction execution.
- An exception refers to any unexpected change in control flow without distinguishing if the cause is internal or external.
- An interrupt is an event that is externally caused.

Event	Source	Terminology
I/O Device Request	External	Interrupt
Syscall	Internal	Exception
Arithmetic Overflow	Internal	Exception
Page Fault	Internal	Exception
Undefined Instruction	Internal	Exception
Hardware Malfunction	Either	Either

MULTIPLE EXCEPTIONS

- Exceptions can occur on different pipeline stages on different instructions.
- Multiple exceptions can occur in the same clock cycle. The load word instruction could have a page fault in the MEM stage and the add instruction could have an integer overflow in the EX stage, both of which are in cycle 4.
- Exceptions can occur out of order. The and instruction could have a page fault in the IF stage (cycle 3), whereas the load word instruction could have a page fault in the MEM stage (cycle 4).

Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
lw	IF	ID	EX	MEM	WB			
add		IF	ID	EX	MEM	WB		
and			IF	ID	EX	MEM	WB	
sub				IF	ID	EX	MEM	WB

PRECISE EXCEPTIONS

Supporting precise exceptions means that:

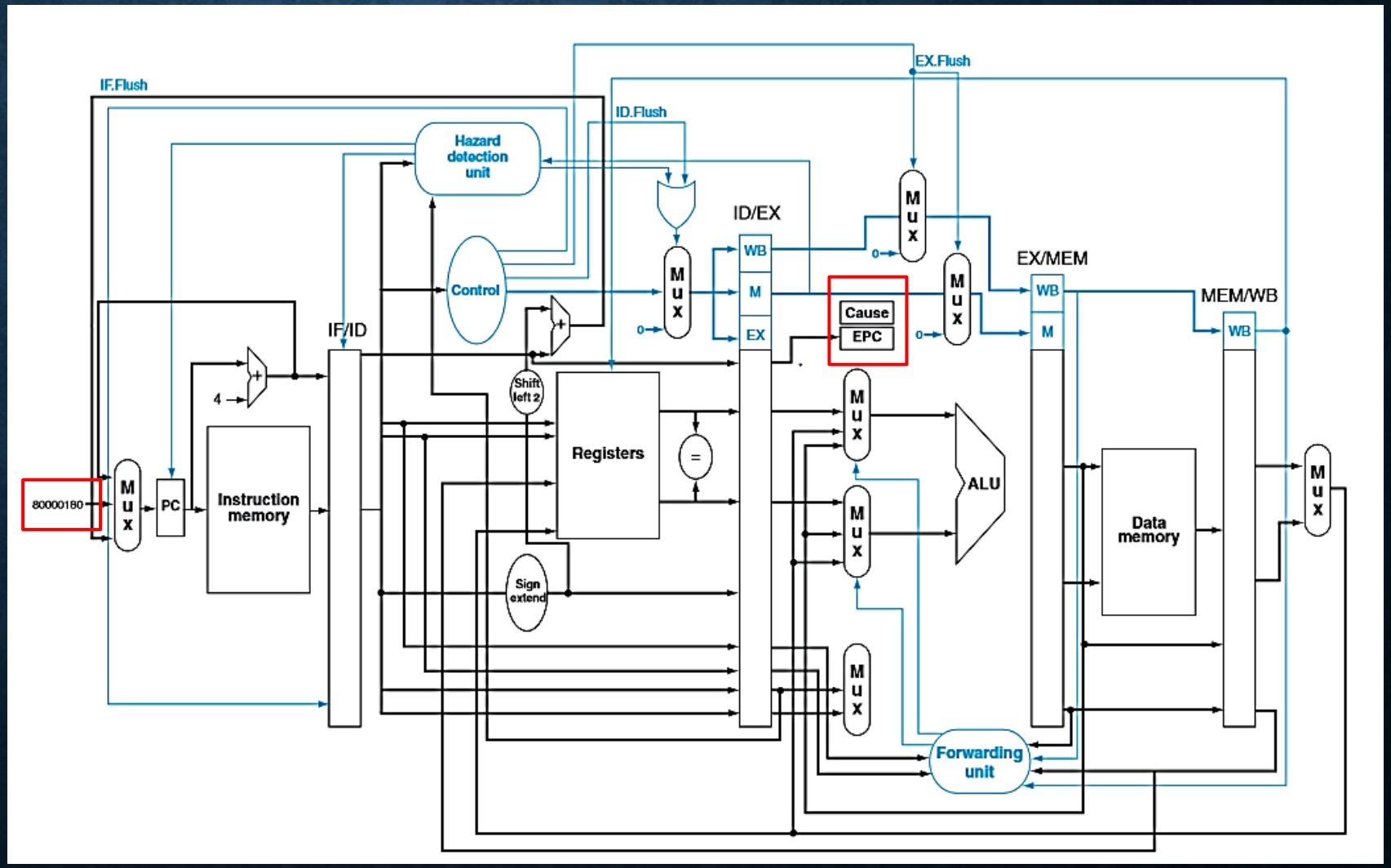
- The exception addressed first is the one associated with the instruction that entered the pipeline first.
- The instructions that entered the pipeline previously are allowed to complete.
- The instruction associated with the exception and any subsequent instructions are flushed.
- The appropriate instruction can be restarted after the exception is handled or the program can be terminated.

HANDLING EXCEPTIONS

- When an exception is detected, the machine:
- Flushes the instructions from the pipeline – this includes the instruction causing the exception and any subsequent instructions.
- Stores the address of the exception-causing instruction in the EPC (Exception Program Counter).
- Begins fetching instructions at the address of the exception handler routine.

DATAPATH WITH EXCEPTION HANDLING

- New input value for PC holds the initial address to fetch instruction from in the event of an exception.
- A Cause register to record the cause of the exception.
- An EPC register to save the address of the instruction to which we should return.



HANDLING AN ARITHMETIC EXCEPTION

- Assume we have the following instruction sequence.

```
40hex   sub   $11, $2, $4
44hex   and   $12, $2, $5
48hex   or    $13, $2, $6
4Chex   add   $1,  $2, $1
50hex   slt   $15, $6, $7
54hex   lw    $16, 50($7)
```

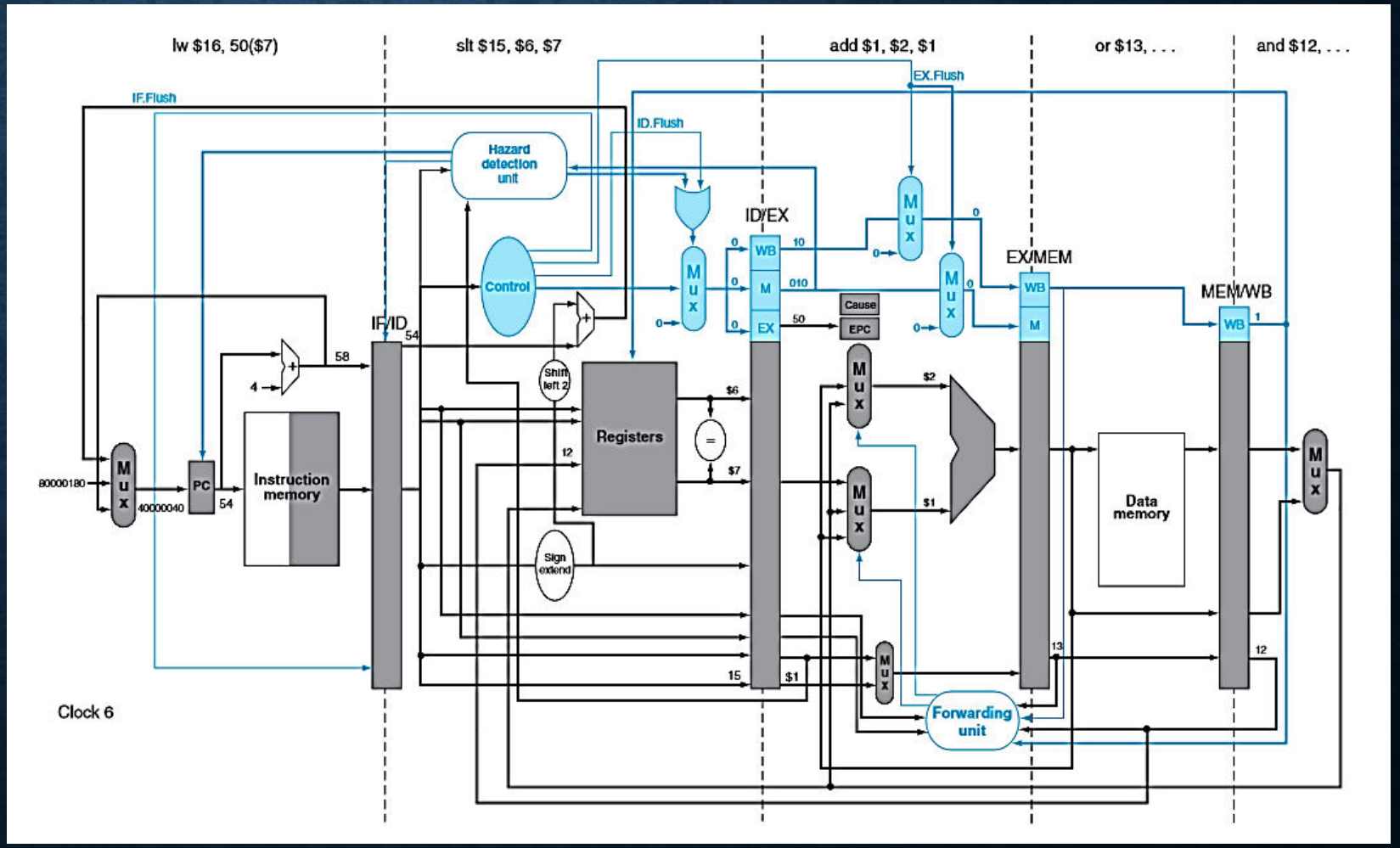
What happens in the pipeline if an overflow exception occurs in the add instruction?

- Also assume that in the event of an exception, the instructions to be evoked begin like this:

```
40000040hex   sw    $25, 1000($0)
40000044hex   sw    $26, 1004($0)
...
```

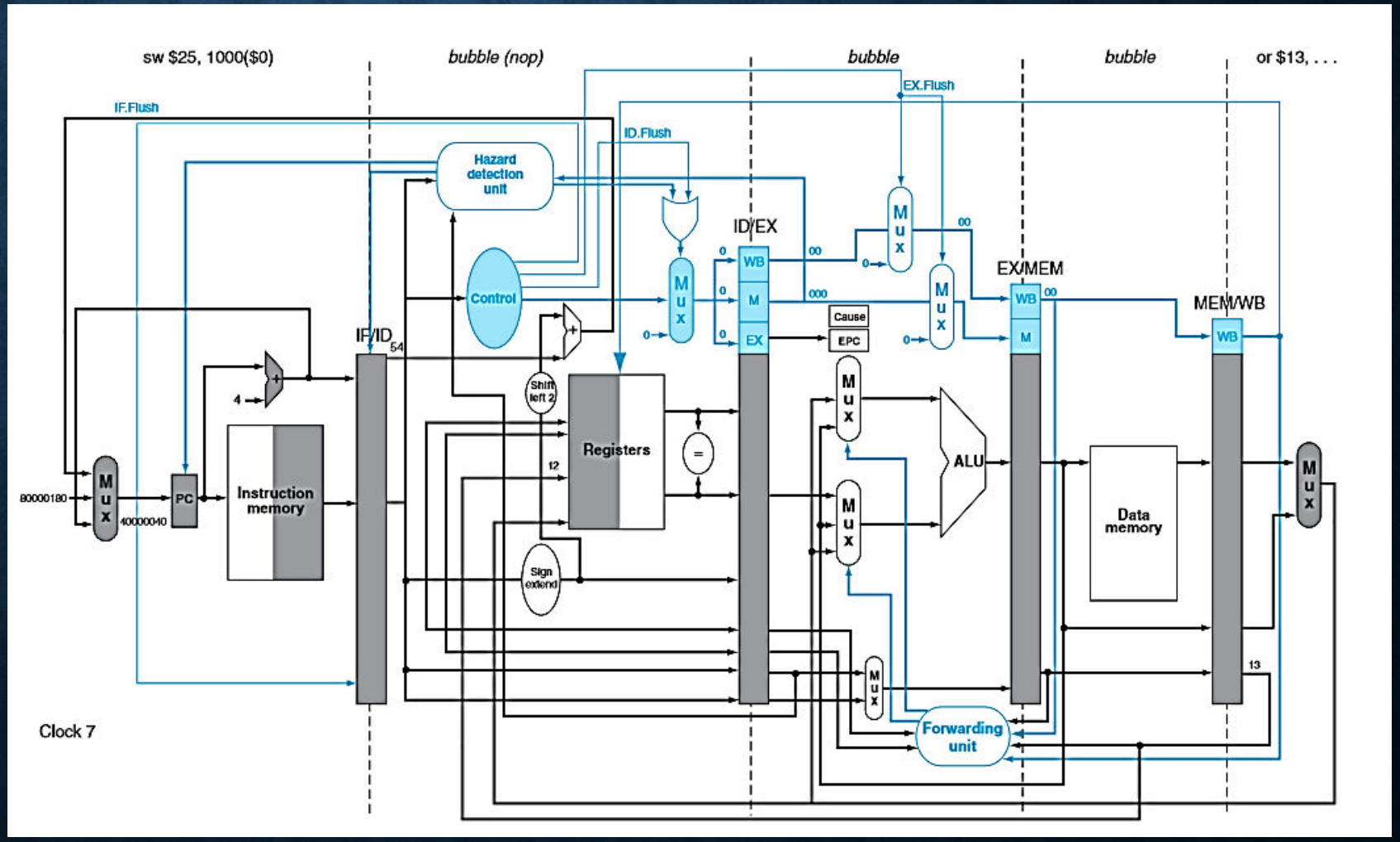
HANDLING AN ARITHMETIC EXCEPTION

- The address after the add is saved in the EPC and flush signals cause control values in the pipeline registers to be cleared.



HANDLING AN ARITHMETIC EXCEPTION

- Instructions are converted into bubbles in the pipeline and the first of the exception handling instructions begins its IF stage.

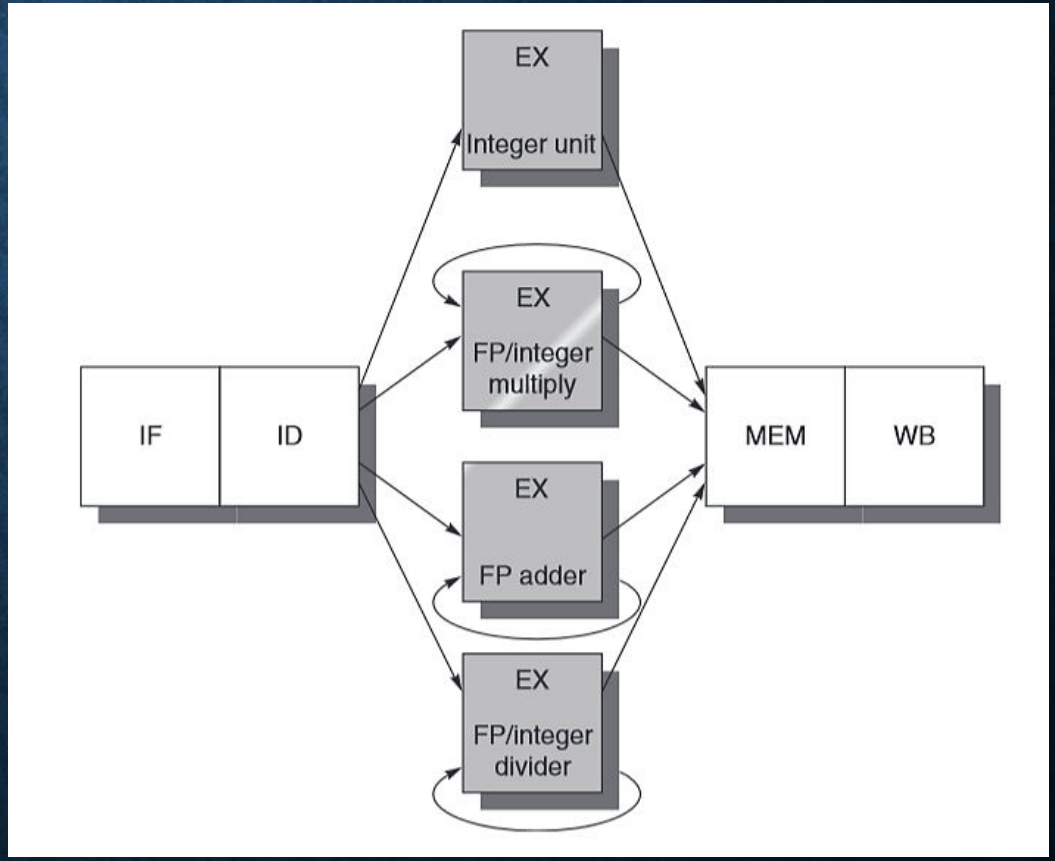


MULTIPLE CYCLE OPERATIONS

- The EX stages of many arithmetic operations are traditionally performed in multiple cycles.
  - integer and floating-point multiplication.
  - integer and floating-point division.
  - floating-point addition, subtraction, and conversions.
- Completing these operations in a single cycle would require a longer clock cycle and/or much more logic in the units that perform these operations.

MULTIPLE CYCLE OPERATIONS

- In this datapath, the multicycle operations loop when they reach the EX stage as these multicycle units are not pipelined. Unpipelined multicycle units can lead to structural hazards.



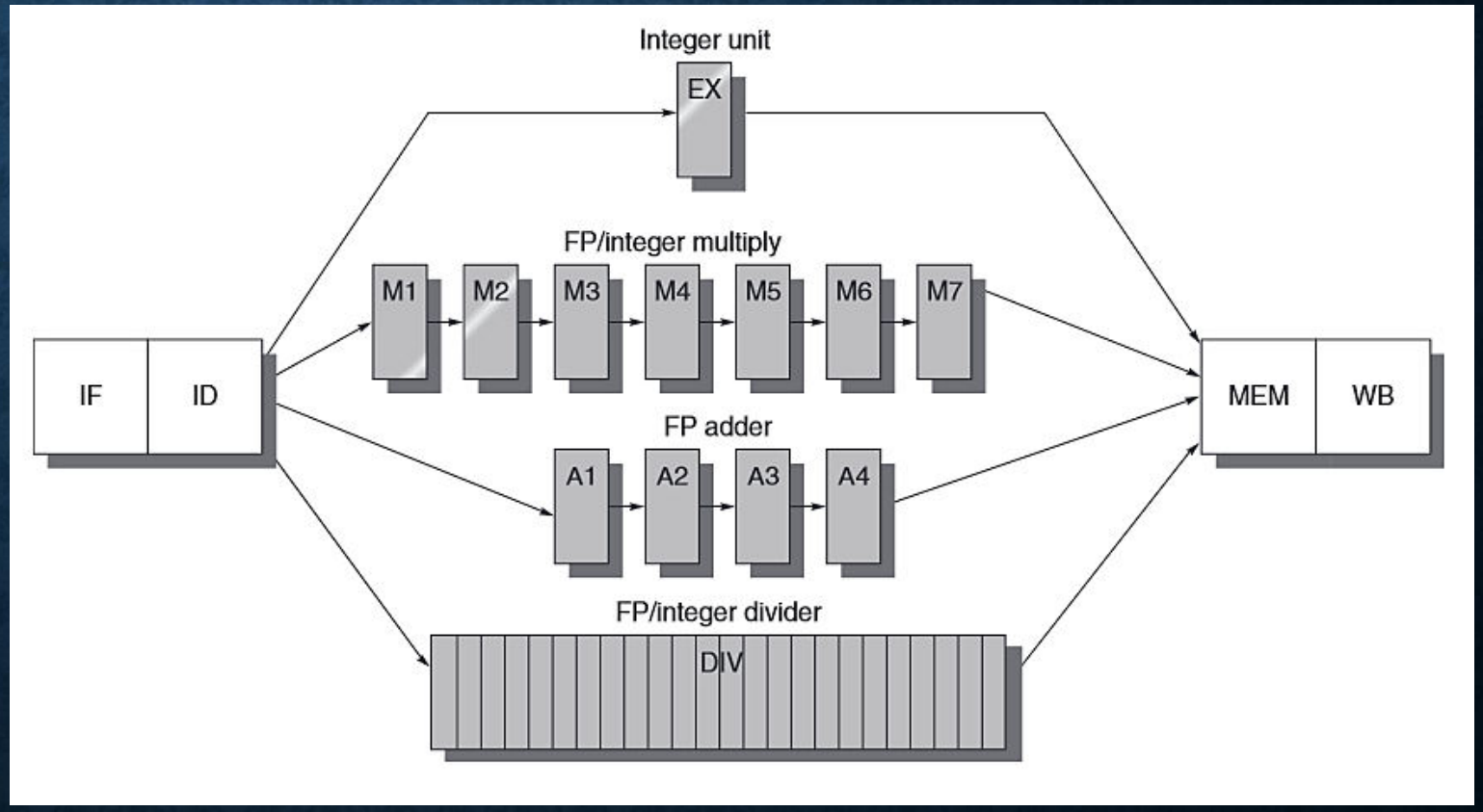
MULTIPLE CYCLE OPERATIONS

- The latency is the minimum number of intervening cycles between an instruction that produces a result and an instruction that uses the result.
- The initiation interval is the number of cycles that must elapse between issuing two operations of a given type.

Functional Unit	Latency	Initiation Interval
Integer ALU	0	1
Data Memory	1	1
FP Add	3	1
FP Multiply	6	1
FP Divide	23	24

MULTIPLE CYCLE OPERATIONS

- The multiplies, FP adds, and FP subtracts are pipelined.
- Divides are not pipelined since this operation is used less often.



MULTIPLE CYCLE OPERATIONS

- Consider this example pipelining of independent (i.e. no dependencies) floating point instructions.
- The states in italics show where data is needed. The states is bold show where data is available.

MUL.D	IF	ID	*M1*	M2	M3	M4	M5	M6	**M7**	MEM	WB
ADD.D		IF	ID	*A1*	A2	A3	**A4**	**MEM**	WB		
L.D			IF	ID	*EX*	**MEM**	WB				
S.D				IF	ID	*EX*	*MEM*	WB			

MULTIPLE CYCLE OPERATIONS

- Stalls for read-after-write hazards will be more frequent.
- The longer the pipeline, the more complicated the stall and forwarding logic becomes.
- Structural hazards can occur when multicycle operations are not fully pipelined.
- Multiple instructions can attempt to write to the FP register file in a single cycle.
- Write-after-write hazards are possible since instructions may not reach the WB stage in order.
- Out of order completion may cause problems with exceptions.

MULTIPLE CYCLE OPERATIONS

- The multiply is stalled due to a load delay.
- The add and store are stalled due to read-after-write FP hazards.

		Clock cycle number																
Instruction		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
L.D	F4,0(R2)	IF	ID	EX	MEM	WB												
MUL.D	F0,F4,F6		IF	ID	Stall	M1	M2	M3	M4	M5	M6	M7	MEM	WB				
ADD.D	F2,F0,F8			IF	Stall	ID	Stall	Stall	Stall	Stall	Stall	Stall	A1	A2	A3	A4	MEM	WB
S.D	F2,0(R2)					IF	Stall	Stall	Stall	Stall	Stall	Stall	ID	EX	Stall	Stall	Stall	MEM

MULTIPLE CYCLE OPERATIONS

- In this example three instructions attempt to simultaneously perform a write-back to the FP register file in clock cycle 11, which causes a write-after-write hazard due to a single FP register file write port. Out of order completion can also lead to imprecise exceptions.

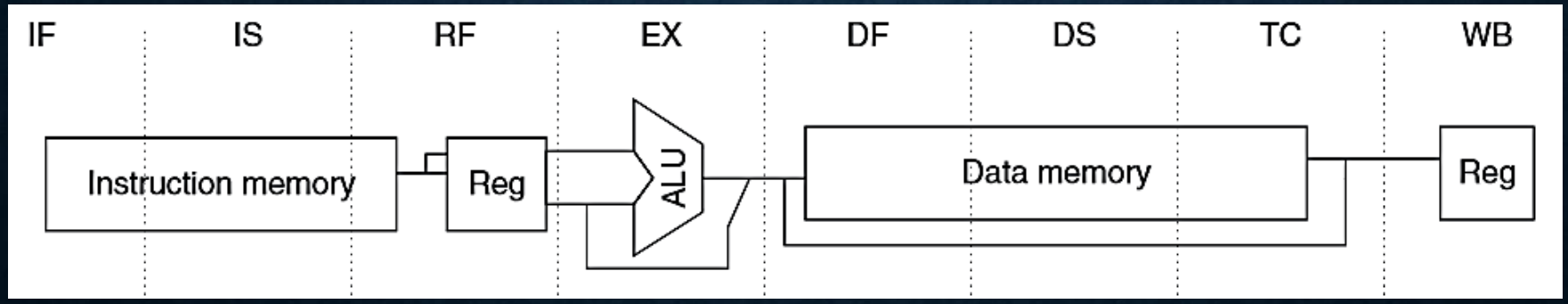
	Clock cycle number										
Instruction	1	2	3	4	5	6	7	8	9	10	11
MUL.D F0,F4,F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
...		IF	ID	EX	MEM	WB					
...			IF	ID	EX	MEM	WB				
ADD.D F2,F4,F6				IF	ID	A1	A2	A3	A4	MEM	WB
...					IF	ID	EX	MEM	WB		
...						IF	ID	EX	MEM	WB	
L.D F2,0(R2)							IF	ID	EX	MEM	WB

MORE INSTRUCTION LEVEL PARALLELISM

- Superpipelining
  - Means more stages in the pipeline.
  - Lowers the cycle time.
  - Increases the number of pipeline stalls.
- Multiple issue
  - Means multiple instructions can simultaneously enter the pipeline and advance to each stage during each cycle.
  - Lowers the cycles per instruction (CPI).
  - Increases the number of pipeline stalls.
- Dynamic scheduling
  - Allows instructions to be executed out of order when instructions that previously entered the pipeline are stalled or require additional cycles.
  - Allows for useful work during some instruction stalls.
  - Often increases cycle time and energy usage.

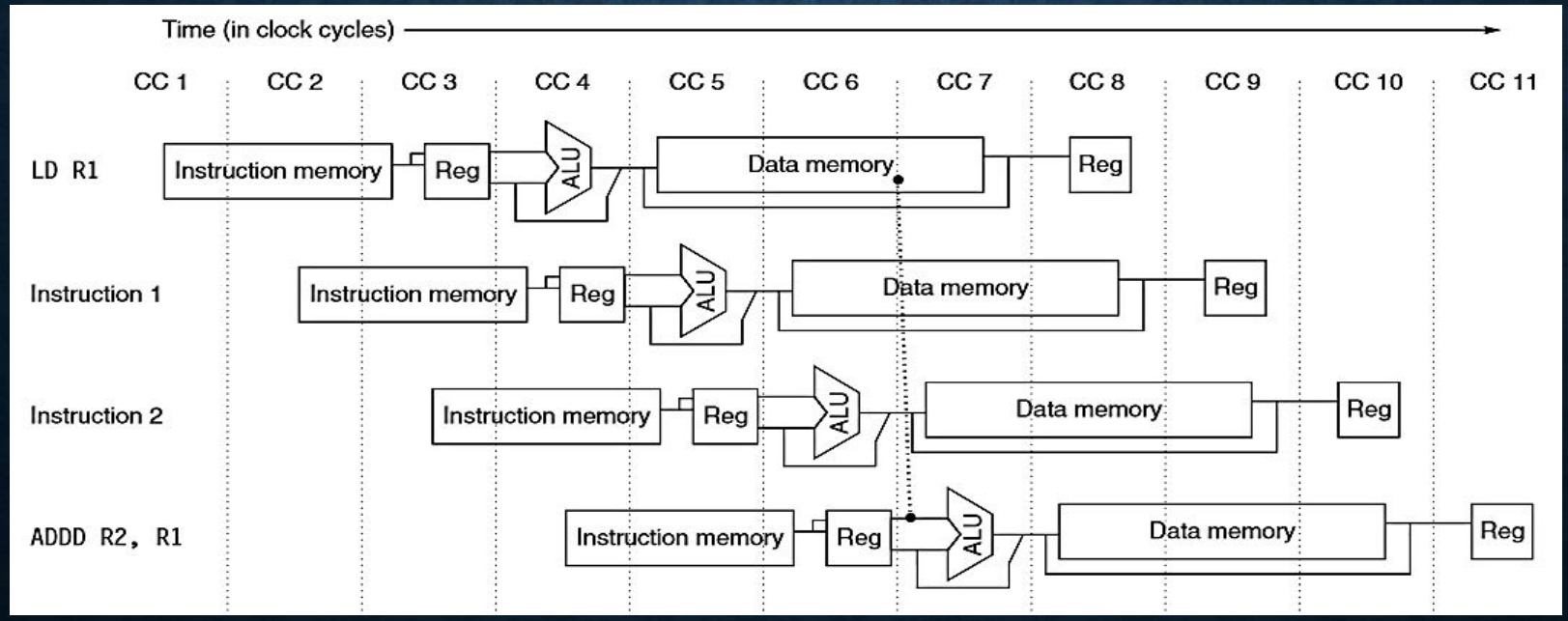
MIPS R4000 PIPELINE

- Below are the stages for the MIPS R4000 integer pipeline.
  - IF - first half of instruction fetch; PC selection occurs here with the initiation of the IC access.
  - IS - second half of instruction fetch; complete IC access.
  - RF - instruction decode, register fetch, hazard checking, IC hit detection.
  - EX - effective address calculation, ALU operation, branch target address calculation and condition evaluation.
  - DF - first half of data cache access.
  - DS - second half of data cache access.
  - TC - tag check to determine if DC access was a hit.
  - WB - write back for loads and register-register operations.



MIPS R4000 PIPELINE

- A two cycle delay is possible because the loaded value is available at the end of the DS stage and can be forwarded.
- If the tag check in the TC stage indicates a miss, then the pipeline is backed up a cycle and the L1 DC miss is serviced.



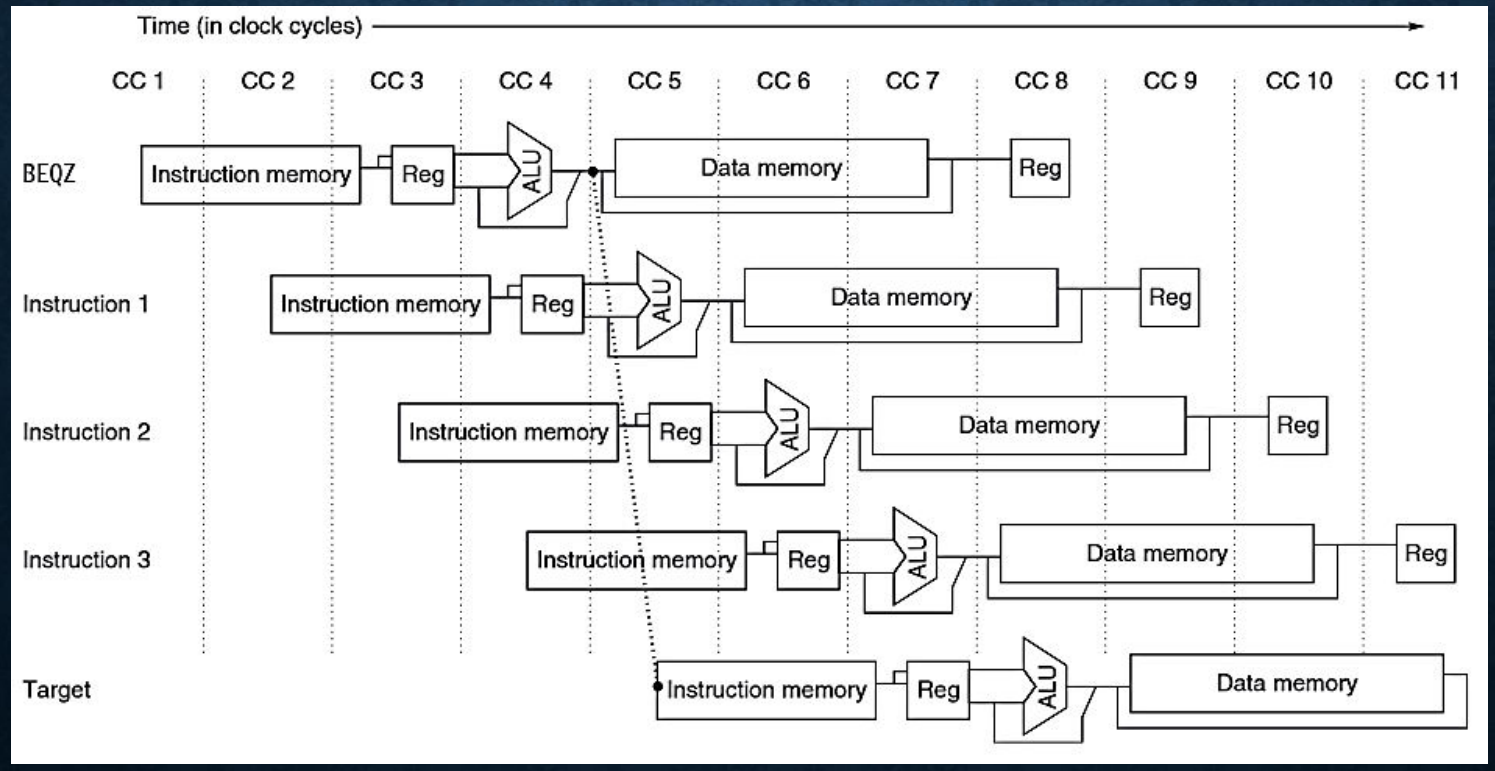
MIPS R4000 PIPELINE

- A load instruction followed by an immediate use of the loaded value results in a 2 cycle stall.

	Clock number								
Instruction number	1	2	3	4	5	6	7	8	9
LD R1,...	IF	IS	RF	EX	DF	DS	TC	WB	
DADD R2,R1,...		IF	IS	RF	Stall	Stall	EX	DF	DS
DSUB R3,R1,...			IF	IS	Stall	Stall	RF	EX	DF
OR R4,R1,...				IF	Stall	Stall	IS	RF	EX

MIPS R4000 PIPELINE

- The branch delay is 3 cycles since the condition evaluation is performed during the EX stage.



MIPS R4000 PIPELINE

- A taken branch on the MIPS R4000 has a 1 cycle delay slot followed by a 2 cycle stall.

instruction	1	2	3	4	5	6	7	8	9	10	11	12
branch instruction	IF	IS	RF	EX	DF	DS	TC	WB				
delay slot		IF	IS	RF	EX	DF	DS	TC	WB			
branch instruction + 2			IF	IS	stall	stall	stall	stall	stall	stall		
branch instruction + 3				IF	stall	stall	stall	stall	stall	stall	stall	
branch target					IF	IS	RF	EX	DF	DS	TC	WB

MIPS R4000 PIPELINE

- A not taken branch on the MIPS R4000 has just a 1 cycle delay slot.

instruction	1	2	3	4	5	6	7	8	9	10	11
branch instruction	IF	IS	RF	EX	DF	DS	TC	WB			
delay slot		IF	IS	RF	EX	DF	DS	TC	WB		
branch instruction + 2			IF	IS	RF	EX	DF	DS	TC	WB	
branch instruction + 3				IF	IS	RF	EX	DF	DS	TC	WB

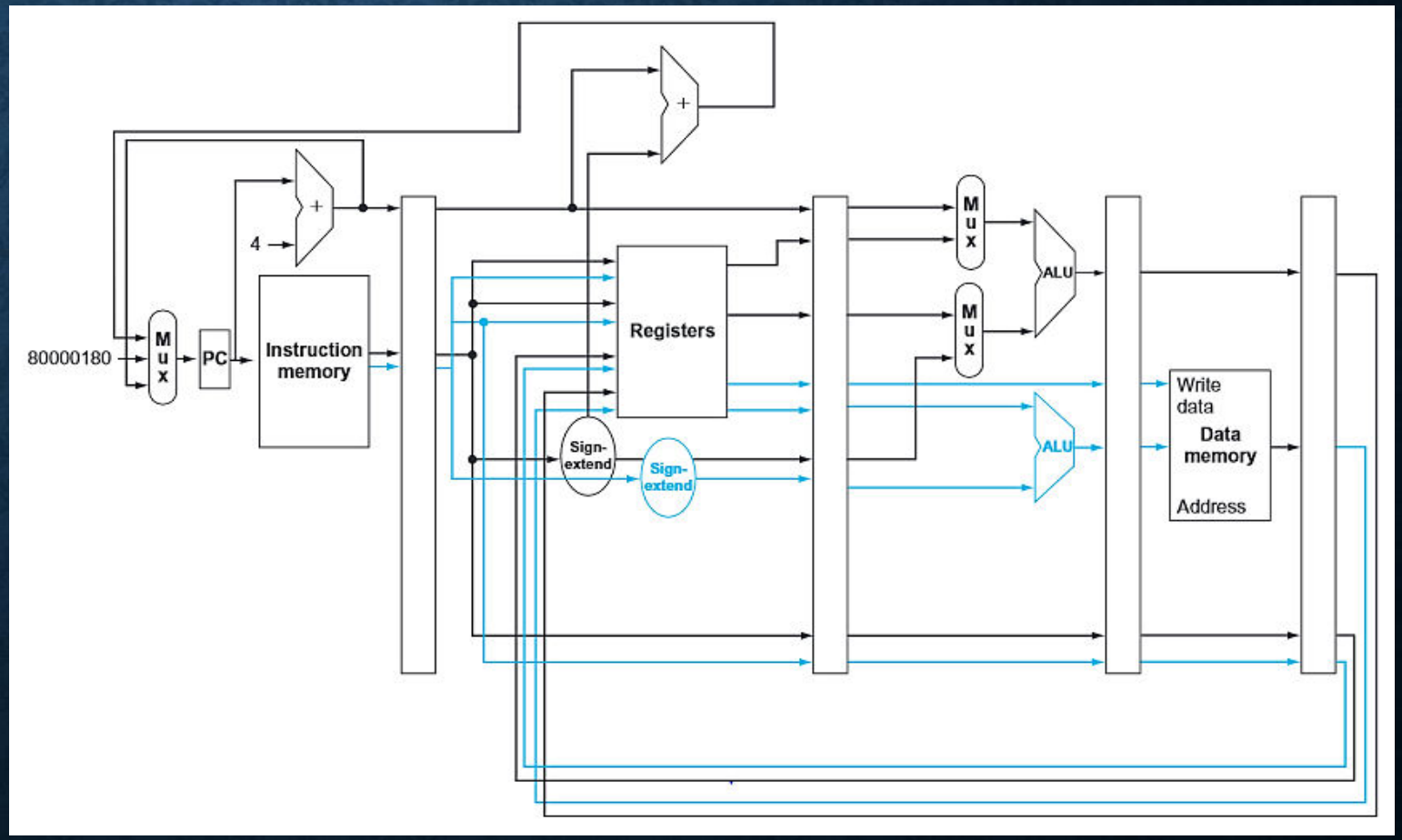
STATIC MULTIPLE ISSUE

- In a static multiple-issue processor, the compiler has the responsibility of arranging the sets of instructions that are independent and can be fetched, decoded, and executed together.
- A static multiple-issue processor that simultaneously issues several independent operations in a single wide instruction is called a Very Long Instruction Word (VLIW) processor. Below is an example static two-issue pipeline in operation.

Instruction type	Pipe stages							
ALU or branch instruction	IF	ID	EX	MEM	WB			
Load or store instruction	IF	ID	EX	MEM	WB			
ALU or branch instruction		IF	ID	EX	MEM	WB		
Load or store instruction		IF	ID	EX	MEM	WB		
ALU or branch instruction			IF	ID	EX	MEM	WB	
Load or store instruction			IF	ID	EX	MEM	WB	
ALU or branch instruction				IF	ID	EX	MEM	WB
Load or store instruction				IF	ID	EX	MEM	WB

STATIC MULTIPLE ISSUE

- The additions needed for double-issue are highlighted in blue.



STATIC MULTIPLE ISSUE

- Original loop in C:

```
for (i = n-1; i != 0; i = i-1)
    a[i] += s
```

- Original loop in MIPS assembly:

```
Loop:  lw $t0,0($s1)     # $t0 = a[i];
       addu $t0,$t0,$s2  # $t0 += s;
       sw $t0,0($s1)     # a[i] = $t0;
       addi $s1,$s1,-4   # i = i-1;
       bne $s1,$zero,Loop # if (i!=0) goto Loop
```

	ALU or branch instruction	Data transfer instruction	Clock cycle
Loop:		lw $t0, 0($s1)	1
	addi $s1,$s1,-4		2
	addu $t0,$t0,$s2		3
	bne $s1,$zero,Loop	sw $t0, 4($s1)	4

DYNAMIC MULTIPLE ISSUE

- Dynamic multiple-issue processors dynamically detect if sequential instructions can be simultaneously issued in the same cycle.
  - no data hazards (dependences)
  - no structural hazards
  - no control hazards
- These type of processors are also known as superscalar.
- One advantage of superscalar over static multiple-issue is that code compiled for single issue will still be able to execute.

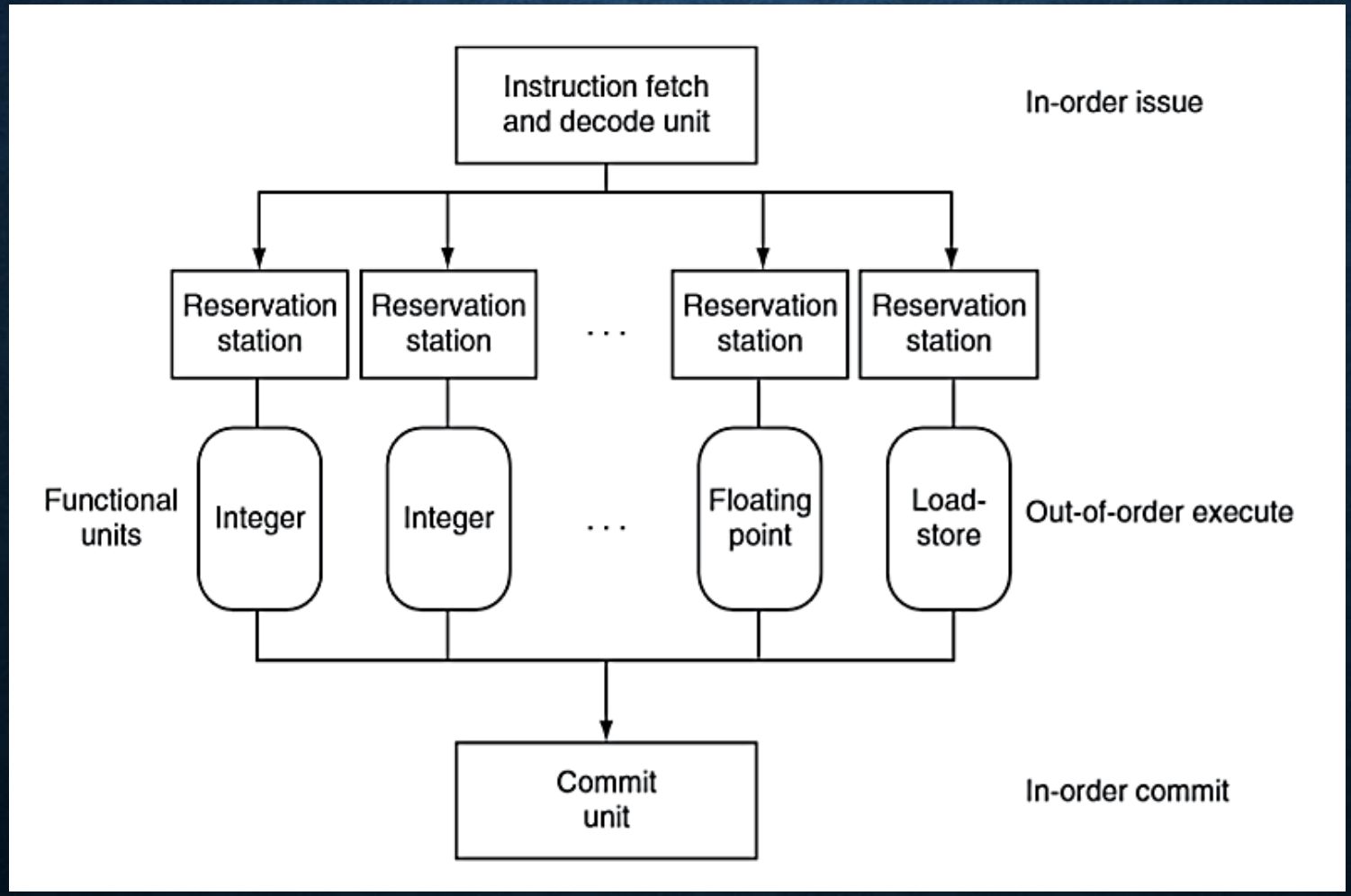
OUT-OF-ORDER EXECUTION PROCESSORS

- Some processors are designed to execute instructions out of order to perform useful work when a given instruction is stalled.
- The add is dependent on the lw, but the sub is independent.

```
lw $1,0($2)
add $3,$4,$1
sub $6,$4,$5
```

- Out-of-order or dynamically scheduled processors:
  - Fetch and issue instructions in order
  - Execute instructions out of order
  - Commit results in order
- Many out-of-order processors also support multi-issue to further improve performance.

DYNAMICALLY SCHEDULED PIPELINE



INTEL MICROPROCESSORS

- Due to thermal limitations, the clock rate has not increased in recent years, which has led to fewer pipeline stages and the adoption of multi-core processors.

Microprocessor	Year	Clock Rate	Pipeline Stages	Issue Width	Out-of-Order/ Speculation	Cores/ Chip	Power	
Intel 486	1989	25 MHz	5	1	No	1	5	W
Intel Pentium	1993	66 MHz	5	2	No	1	10	W
Intel Pentium Pro	1997	200 MHz	10	3	Yes	1	29	W
Intel Pentium 4 Willamette	2001	2000 MHz	22	3	Yes	1	75	W
Intel Pentium 4 Prescott	2004	3600 MHz	31	3	Yes	1	103	W
Intel Core	2006	2930 MHz	14	4	Yes	2	75	W
Intel Core i5 Nehalem	2010	3300 MHz	14	4	Yes	1	87	W
Intel Core i5 Ivy Bridge	2012	3400 MHz	14	4	Yes	8	77	W

EMBEDDED AND SERVER PROCESSORS

Processor	ARM A8	Intel Core i7 920
Market	Personal Mobile Device	Server, Cloud
Thermal design power	2 Watts	130 Watts
Clock rate	1 GHz	2.66 GHz
Cores/Chip	1	4
Floating point?	No	Yes
Multiple Issue?	Dynamic	Dynamic
Peak instructions/clock cycle	2	4
Pipeline Stages	14	14
Pipeline schedule	Static In-order	Dynamic Out-of-order with Speculation
Branch prediction	2-level	2-level
1st level caches / core	32 KiB I, 32 KiB D	32 KiB I, 32 KiB D
2nd level cache / core	128–1024 KiB	256 KiB
3rd level cache (shared)	–	2–8 MiB