ISA

SC6231 (Advanced Computer Architecture)

Student : Abdulle Hassan

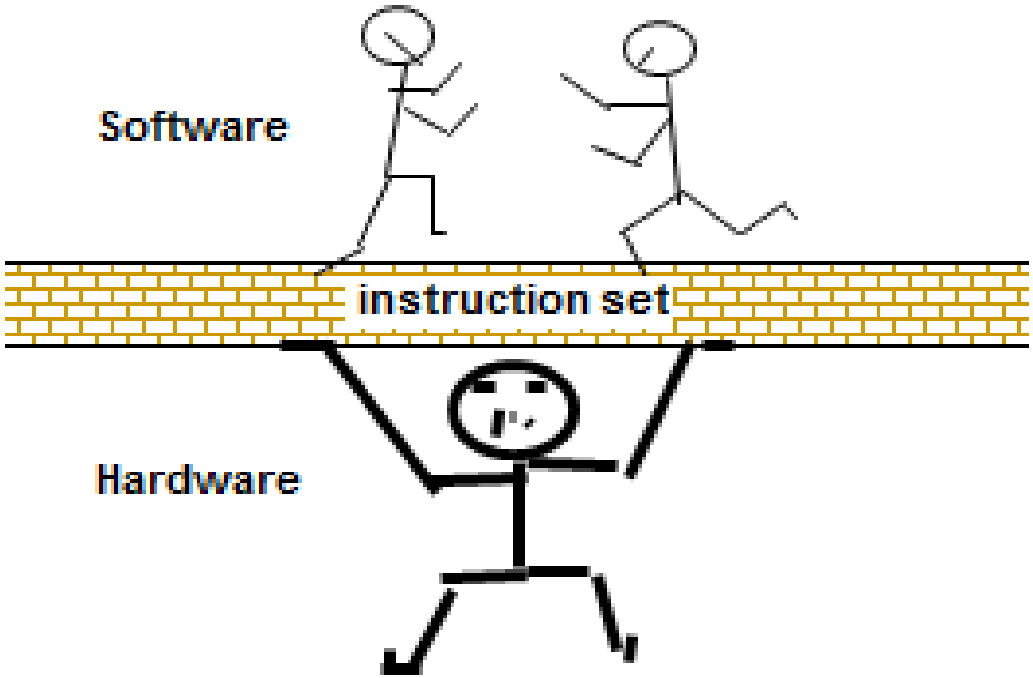
ID: g6029694

Agenda

1. Definitions
2. Simulations
3. Discussions

Definitions

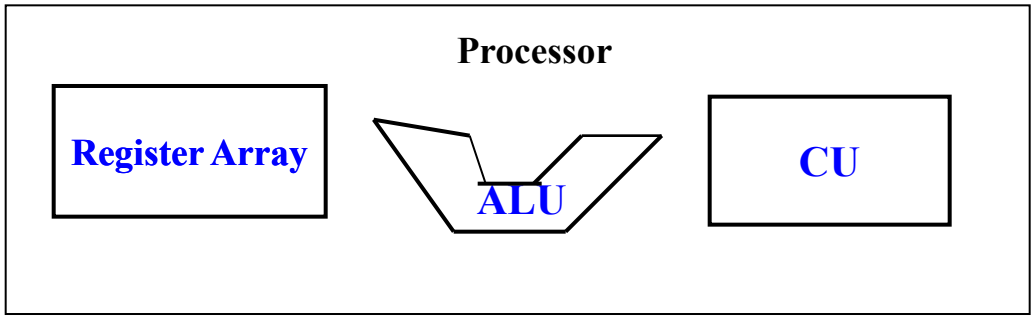
ISA:
The portion of the computer visible to the programmers or compiler writers

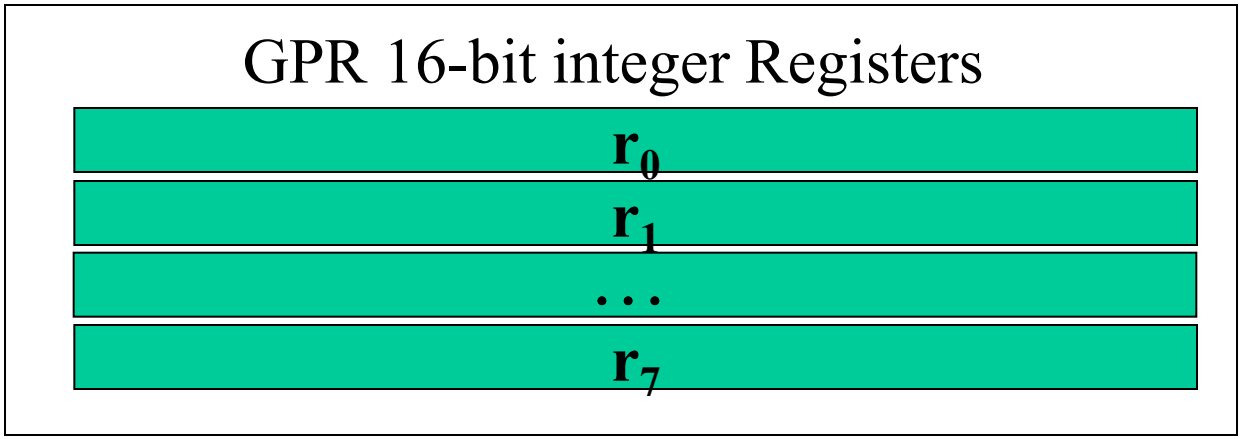


ISA Design

Class of architecture:

16-bit processor based ISA



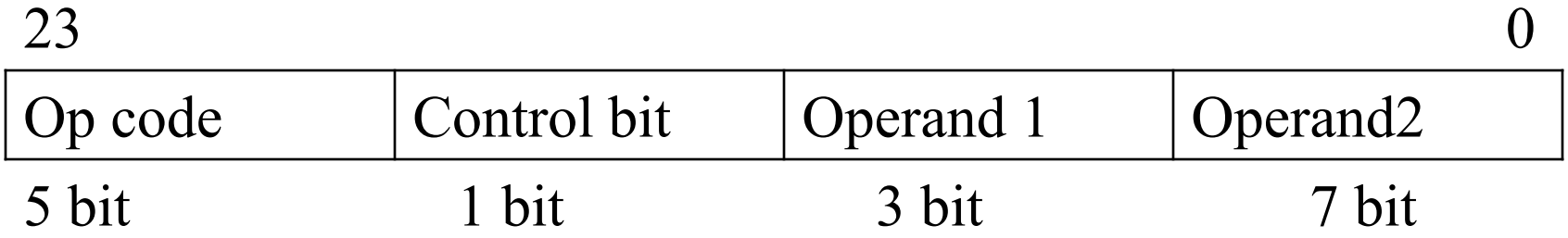


ISA Design

- This ISA uses sixteen 16-bit General Purpose Registers (r0 – r7), and two additional registers of size 32 bit size registers called ‘rm’ to store the result of multiplication operation and register ‘re’ to store the division operation.

Memory Design

Byte addressable 24-bit Memory design



The memory design can address up to 16MB memory location any one time

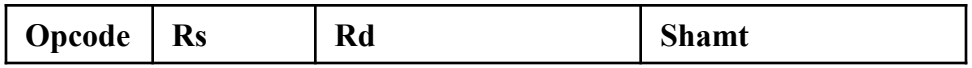
Memory Design

The ISA GPR registers are of 16 bit length.

The designed memory of the Instruction Set Architecture is of size 24-bit architecture.

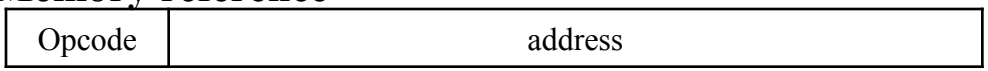
Basic Instruction Format

Register-Register



Register-Immediate

Memory reference



Basic Instruction Format

Operation Encoding

OpCode	Encoding
Arithmetic Instructions	
Add	00010
Add Imm	00011
Sub	00101
Mul	00111
Mul Imm	00111
Div	01000
Div Imm	01001
Data Transfer Instructions	
Mov	00001

Encoding Instructions

Op Code	Control Bit	Operations	Addressing Modes
00000	0	MOV	Register Addressing Mode
00001	1		Immediate Addressing Mode
00010	0	ADD	Register Address Mode
00011	1		Immediate Addressing Mode
00101	0	SUB	Register Address Mode
00101	1		Immediate Addressing Mode
00110	0	MUL	Register Address Mode
00111	1		Immediate Addressing Mode
01000	0	DIV	Register Address Mode
01001	1		Immediate Addressing Mode

Encoding Summary

MOV

1. Copies an integer value into one of the GPR
2. Copies a register content into another GPR

Example

- MOV r0 r1 (moves r1 contents into r0
- MOV r0 10 (moves constant value10 into r0

ADD

1. Sums two values stored in two GPRs and store the result into the destination one
2. Sums Constant values and content of GPR 1ans store the result into the Destinations

Example

- ADD r0 r1 // r0 = r0 + r1
- ADD R0 AFH // r0 = r0 + AFH

SUB

Subtracts two values stored in two GPR registers,

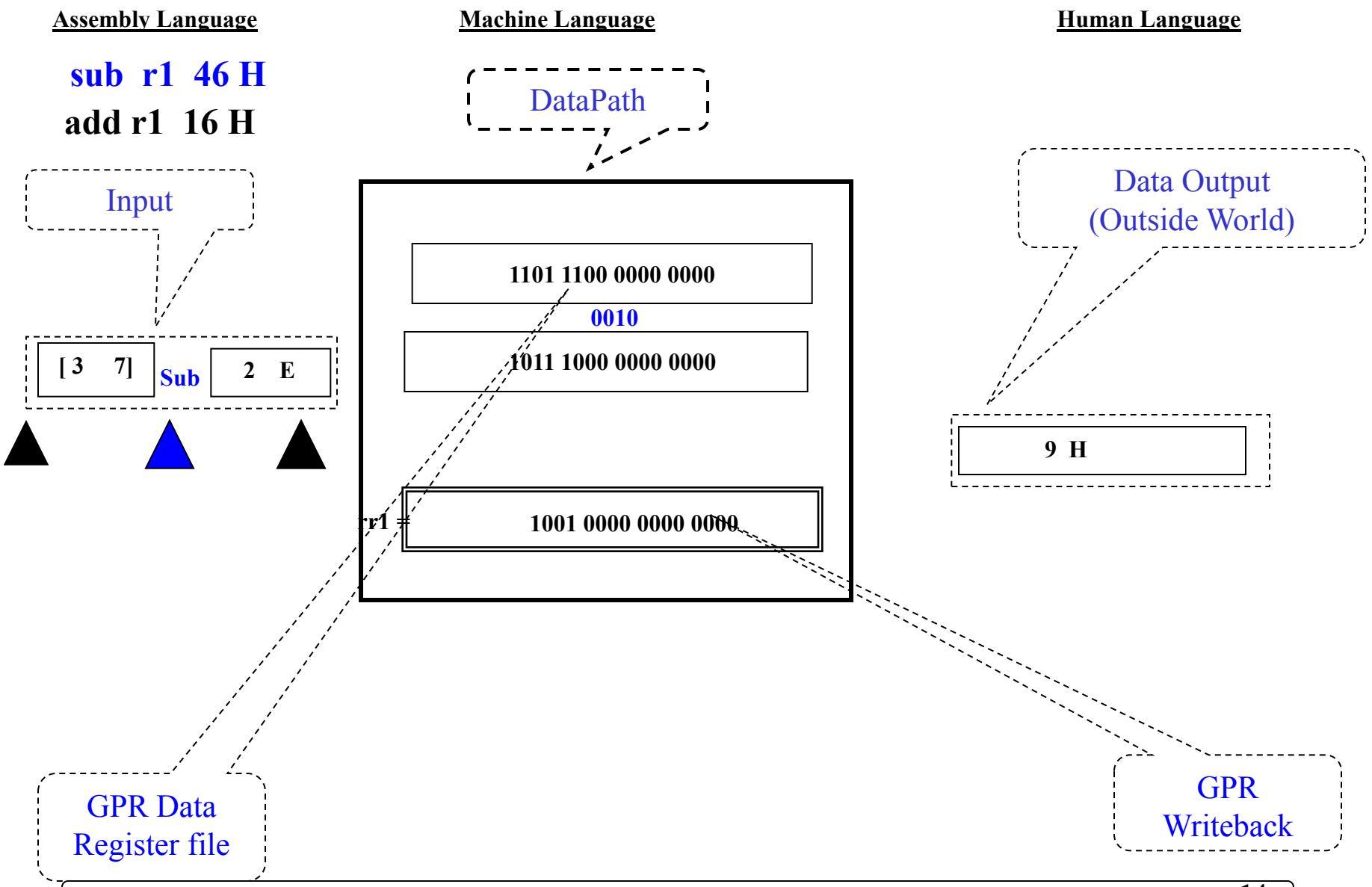
Subtracts integer value from a value stored in the register.

Example

SUB r0 r1 // r0 = r0 + Comp(r1) +1

SUB r0 FA H // r0 = r0 + 05H +1

Virtual Processor Unpipelined,
ISA Format of I-Type



Multiple implementations for a single architecture:

Multiple implementations for a single architecture:

- **Single-cycle implementation**
  - A technique in which an instruction is executed in one clock cycle [1].

Multiple implementations for a single architecture:

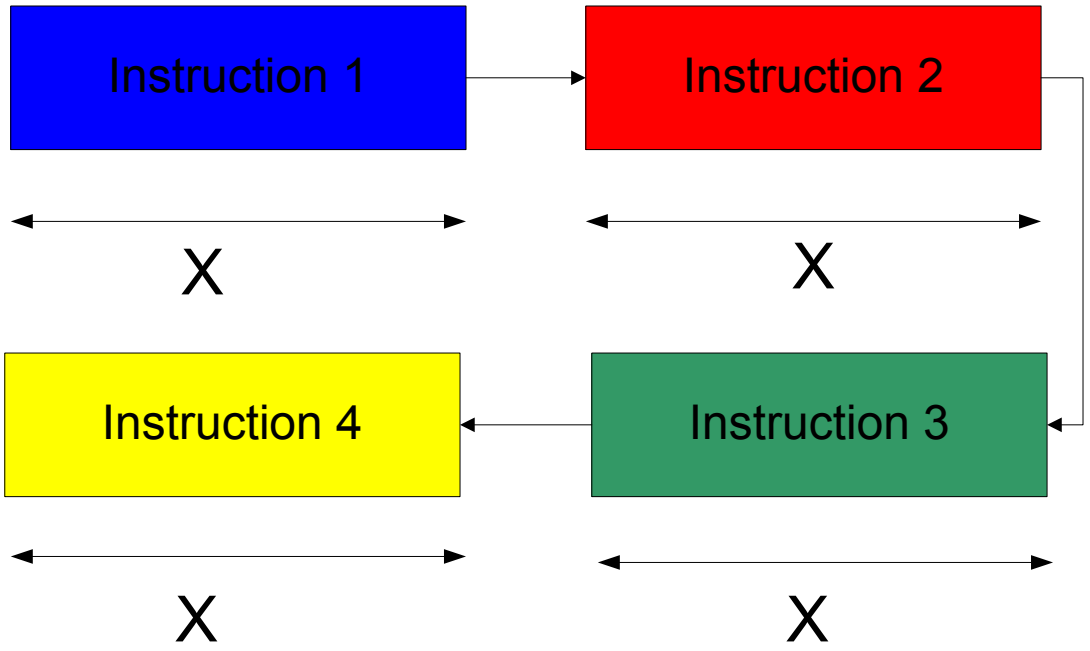
- **Multi-cycle implementation**
  - A technique in which an instruction is executed in multiple clock cycles [1, 3].

Multiple implementations for a single architecture:

- **Pipelining implementation**
  - A technique that exploits parallelism among the instructions in a sequential instruction stream [1, 2].

Single-cycle implementation

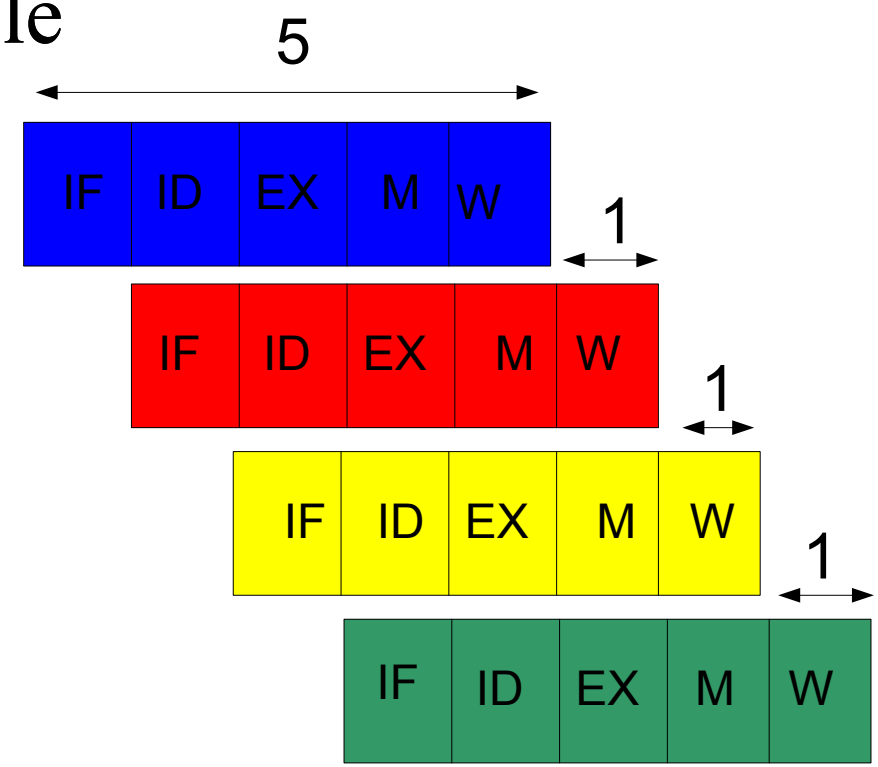
Example



Four sample instructions, executed linearly

Pipelined implementation

Example



Four Pipelined Instructions

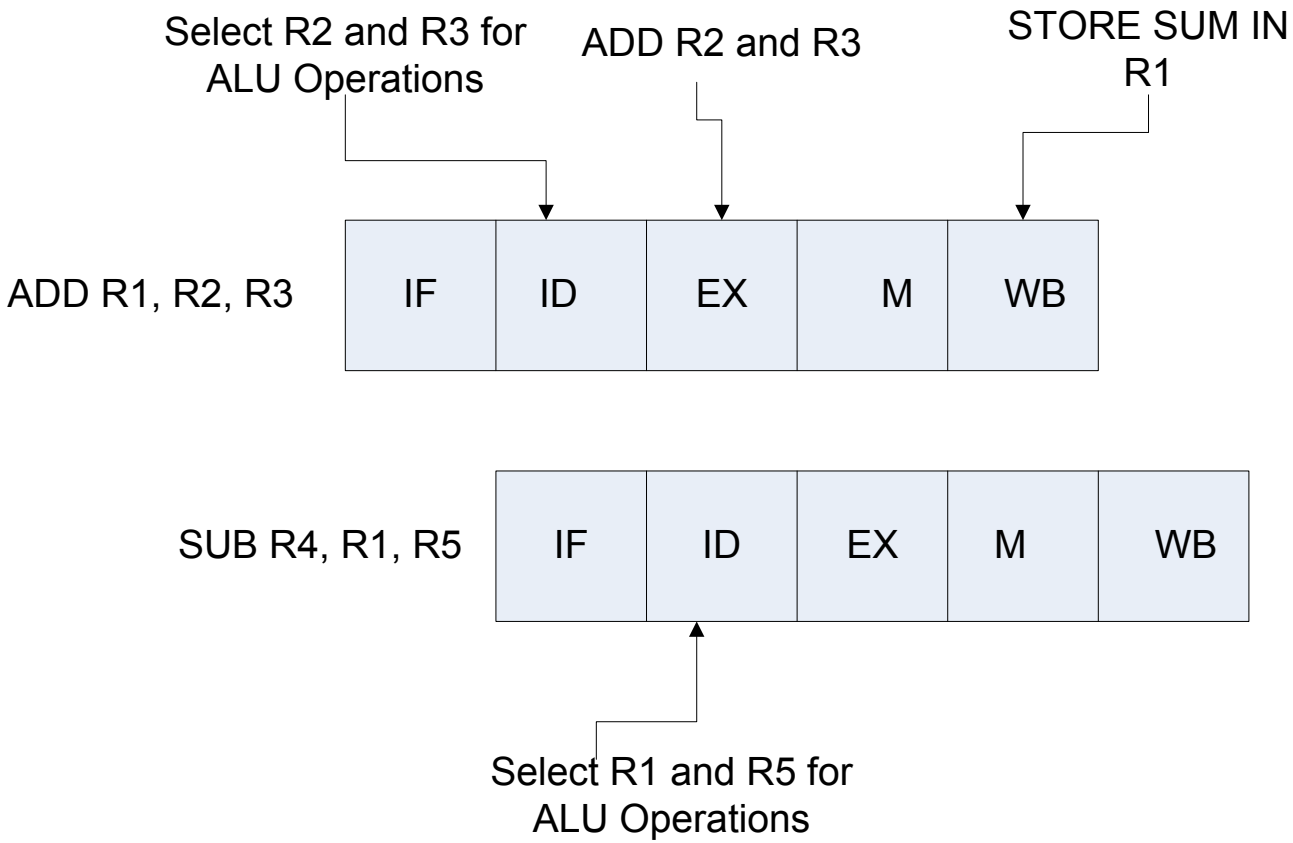
Pipeline Hazards

- Data Hazards – an instruction uses the result of the previous instruction. A hazard occurs exactly when an instruction tries to read a register in its ID stage that an earlier instruction intends to write in its WB stage.

- Control Hazards – the location of an instruction depends on previous instruction

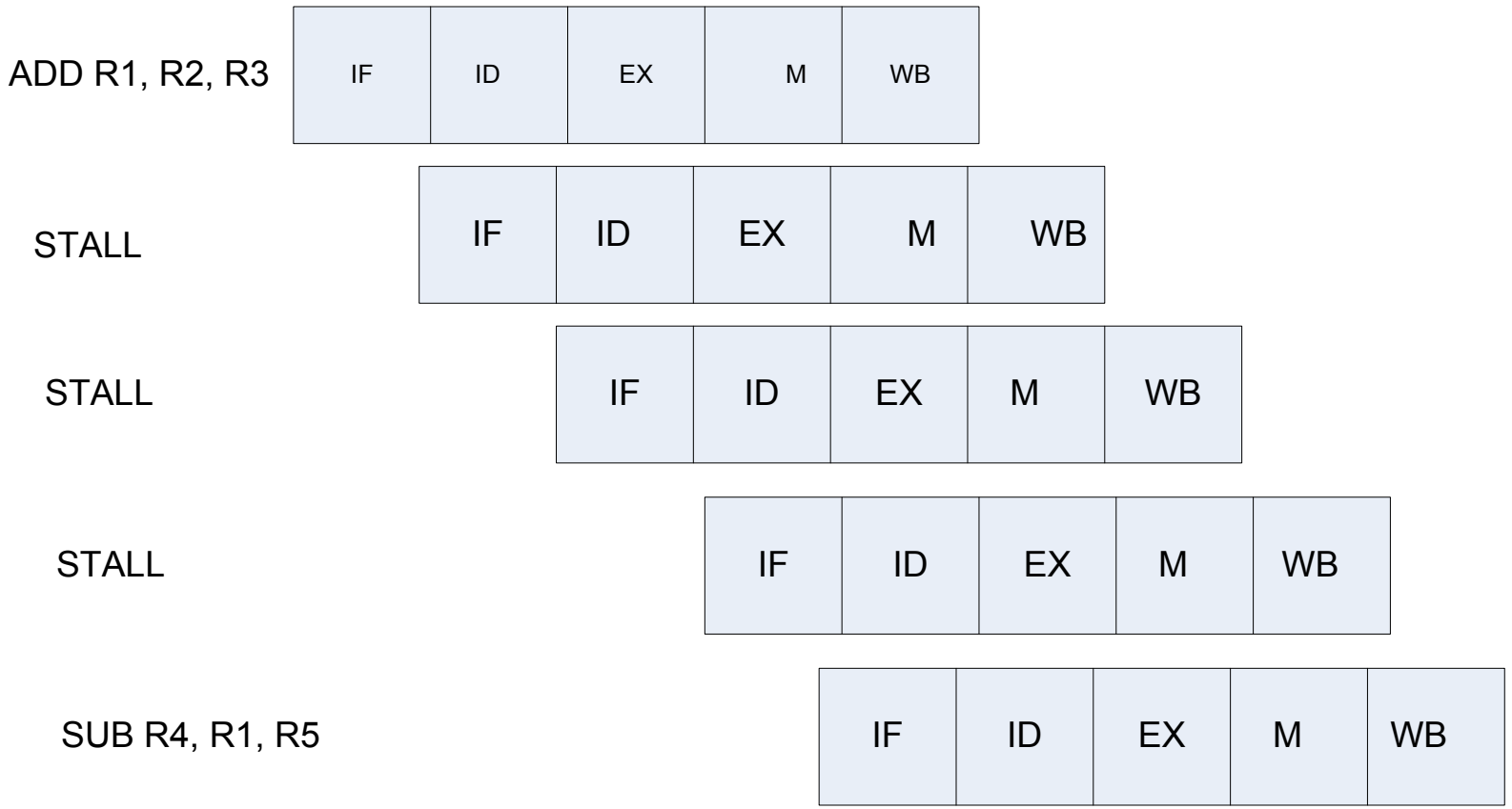
- Structural Hazards – two instructions need to access the same resource

Data Hazards



Stalling

- Stalling involves halting the flow of instructions until the required result is ready to be used. However stalling wastes processor time by doing nothing while waiting for the result.


ADD R1, R2, R3
IF
ID
EX
M
WB
STALL
IF
ID
EX
M
WB
STALL
IF
ID
EX
M
WB
STALL
IF
ID
EX
M
WB
SUB R4, R1, R5
IF
ID
EX
M
WB

SIMULATIONS

Pipeline Examples

```
--------------------------------

INTEL BASED ISA ARCHITECTURE
Course name: SC6231-2-2017 CA
Student: Abdulle Hassan
ID: g6029694
--------------------------------



Press any key to proceed    :
```

Pipeline Examples

```
SPECIFICATION of the ISA
-------------------------------------------------------------------
 * Memory Addressable              : 16MB

 * ALU,GPRs and all instructions-
   are designed to work with       : 16 bit binary words.

 * Instructin Set                  : 24 bits

 * Registers                       : 8 of 16 bits(r0 to r7)

 * CPI                             : One Clock cycle + PIPELINE delay

 * PIPELINE Data Hazards           : via Forwarding

 * ALU Unsigned 2's complement     : Enabled with 16 bit Sign Ext.
-------------------------------------------------------------------

Press any key to continue...
```

Pipeline Examples

```
Welcome to a user designed INTEL Based 24 bit ISA:
 Instructions==========================================================================
 1.      Select the Mnemonics like 'mov',  'div',
         'add', 'sub', 'mul',or 'end' to terminate the program>:

 2.      Select freely the first register <r0, r1, r2, r3,.... or r7>
 3.      Again select freely the second register from remaining <r0,.....,r7> or
         a decimal value>


 For example,'mov r1 48' or  'mov r2 r1' or type 'end 0 0' to terminate.
 ==========================================================================

 Press any key to launch....:
```

Pipeline Examples

```
Enter statemets or end 0 0 to finish
 mov r1 77
 end 0 0

      PC Decoded: Encoded instructions(24-bit): Clock cycles

            PC[0]-> mov r1, 77: 00001 001    0000000001001101        1


      After the program execution contents of the regiters are.......


      r1 = 77 [0000000001001101]

      CPI of the program..........

      CPI =  1.00
      Note: There are only 1 instructions in the program.


               Pipelined Execution of the Program
               ==================================================================
               CPU with 5-pipeline stages: IF | ID | EX | MEM (Load)| WB (Write back)
               Duration of each phase: one clock cycle
               RAW Hazard if any will be detected
               Data RAW Hazard is resolved via FORWARDING.
              ==================================================================
                                  1   2   3   4   5   6   7   8   9   10
               1. mov r1 77 :                 IF | ID | EX | MEM | WB


               RAW hazard details:
               ====================

               There is no RAW hazard detected.
```

Pipeline Examples

```
Enter statemets or end 0 0 to finish
 mov r1 32
 mov r2 68
 add r2 r1
 add r2 100
 add r1 168
 sub r2 r1
 end 0 0
```

```
PC Decoded: Encoded instructions(24-bit): Clock cycles

      PC[0]-> mov r1, 32: 00001 001   0000000000100000       1
      PC[1]-> mov r2, 68: 00001 010   0000000001000100       1
      PC[2]-> add r2, r1: 00010 010   0010000000000000       2
      PC[3]-> add r2, 100: 00011 010   0000000001100100       2
      PC[4]-> add r1, 168: 00011 001   0000000010101000       2
      PC[5]-> sub r2, r1: 00100 010   0010000000000000       3


After the program execution contents of the regiters are........


r1 = 32 [0000000000100000]
r2 = 68 [0000000001000100]
r2 = 68 [0000000001000100]
r2 = 168 [0000000010101000]
r1 = 200 [0000000011001000]

CPI of the program...........

CPI =  1.83
Note: There are only 6 instructions in the program.
```

Pipeline Examples

```
Enter statemets or end 0 0 to finish
 mov r1 32
 mov r2 68
 add r2 r1
 add r2 100
 add r1 168
 sub r2 r1
 end 0 0

 Pipelined Execution of the Program
 ==================================================================================
 CPU with 5-pipeline stages: IF | ID | EX | MEM (Load)| WB (Write back) .
 Duration of each phase: one clock cycle
 RAW Hazard if any will be detected
 Data RAW Hazard is resolved via FORWARDING.
==================================================================================
                         1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
 1. mov r1 32 :                          IF | ID | EX | MEM | WB
 2. mov r2 68 :                               IF | ID | EX | MEM | WB
 3. add r2 r1 :                                    IF | ID | EX | MEM | WB
 23. add r2 100 :                                        IF | ID | EX | MEM | WB
 24. add r1 168 :                                             IF | ID | EX | MEM | WB
 6. sub r2 r1 :                                                    IF | ID | EX | MEM | WB


 RAW hazard details:
 ====================
 r1 in instructions 5 and 6 caused RAW hazard and resolved via Forwarding.



 The CPU uses 10 clock cycles for its pipeline execution.

 Press any key to continue or 'n' to exit the simulation :
```

References

1. Computer Architecture: A Quantitative Approach, John L. Hennessy and David A. Patterson, Morgan Kaufmann, 20016 ISBN: 9716-161-312-0726-0
2. Computer architecture and implementation (b-ok.org) Harvey G Cragon , Pearson, 2010.
3. Instruction Set Architecture Impact on Design Space Subsetting for Configurable Systems, Mohamad Hammam Alsafrjalani, 2017 IEEE 3rd International Proc. Pp 1-5.

Thank you

Discussions