Project

General Description

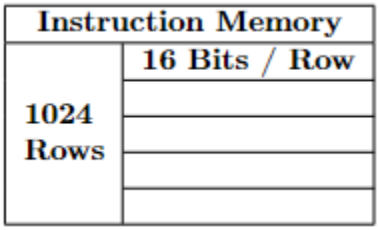
Package 2: Double McHarvard with cheese circular shifts

Memory Architecture

a) Architecture: Harvard

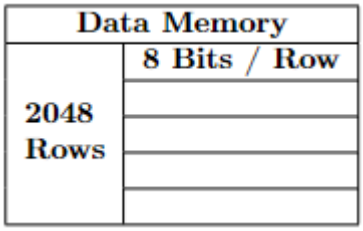
• Harvard Architecture is the digital computer architecture whose design is based on the concept where there are separate storage and separate buses (signal path) for instruction and data. It was basically developed to overcome the bottleneck of Von Neumann Architecture.

b) Instruction Memory Size: 1024 * 16



- The instruction memory addresses are from 0 to 2^10 − 1 (0 to 1023).
- Each memory block (row) contains 1 word which is 16 bits (2 bytes).
- The instruction memory is word addressable.
- The program instructions are stored in the instruction memory

c) Data Memory Size: 2048 * 8



• The data memory addresses are from 0 to 2^11 − 1 (0 to 2047).
• Each memory block (row) contains 1 word which is 8 bits (1 byte).
• The data memory is word/byte addressable (1 word = 1 byte).
• The data is stored in the data memory.

d) Registers: 66

• Size: 8 bits
• 64 General-Purpose Registers (GPRS)
– Names: R0 to R63
• 1 Status Register

7	6	5	4	3	2	1	0
0	0	0	C	V	N	S	Z

– Name: SREG
– A status register, flag register, or condition code register (CCR) is a collection of status flag bits for a processor.
– The status register has 5 flags updated after the execution of specific instructions:

∗ Carry Flag (C): Indicates when an arithmetic carry or borrow has been generated out of the most significant bit position.
· Check on 9th bit (bit 8) of UNSIGNED[VALUE1] OP UNSIGNED[VALUE2] ==1 or not.
∗ Two's Complement Overflow Flag (V): Indicates when the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit.

Project

General Description

· If 2 numbers are added, and they both have the same sign (both positive or both negative), then overflow occurs (V = 1) if and only if the result has the opposite sign. Overflow never occurs when adding operands with different signs.

· If 2 numbers are subtracted, and their signs are different, then overflow occurs (V=1) if and only if the result has the same sign as the subtrahend.

∗ Negative Flag (N): Indicates a negative result in an arithmetic or logic operation.
· N = 1 if result is negative.
· N = 0 if result is positive or zero.

∗ Sign Flag (S): Indicates the expected sign of the result (not the actual sign).
· S = N ⊕ V (XORing the negative and overflow flags will calculate the sign flag).

∗ Zero Flag (Z): Indicates that the result of an arithmetic or logical operation was zero.
· Z = 1 if result is 0.
· Z = 0 if result is not 0.

∗ Since all registers are 8 bits, and we are only using 5 bits in the Status Register for the flags, you are required to keep Bits7:5 cleared “0” at all times in the register.

• 1 Program Counter

– Name: PC
– Type: Special-purpose register with a size of 16 bits (not 8 bits).
– A program counter is a register in a computer processor that contains the address(location) of the instruction being executed at the current time.
– As each instruction gets fetched, the program counter is incremented to point to the next instruction to be executed.

Instruction Set Architecture

a) Instruction Size: 16 bits

b) Instruction Types: 2

R-Format		
OPCODE	R1	R2
4	6	6

I-Format		
OPCODE	R1	IMMEDIATE
4	6	6

c) Instruction Count: 12

• The opcodes are from 0 to 11 according to the instructions order in the following table:

Name	Mnemonic	Type	Format	Operation
Add	ADD	R	ADD R1 R2	R1 = R1 + R2
Subtract	SUB	R	SUB R1 R2	R1 = R1 - R2
Multiply	MUL	R	MUL R1 R2	R1 = R1 * R2
Load Immediate	LDI	I	LDI R1 IMM	R1 = IMM
Branch if Equal Zero	BEQZ	I	BEQZ R1 IMM	IF(R1 == 0) { PC = PC+1+IMM }
And	AND	R	AND R1 R2	R1 = R1 & R2
Or	OR	R	OR R1 R2	R1 = R1 | R2
Jump Register	JR	R	JR R1 R2	PC = R1 || R2
Shift Left Circular	SLC	I	SLC R1 IMM	R1 = R1 << IMM | R1 >>> 8 - IMM
Shift Right Circular	SRC	I	SRC R1 IMM	R1 = R1 >>> IMM | R1 << 8 - IMM
Load Byte	LB	I	LB R1 ADDRESS	R1 = MEM[ADDRESS]
Store Byte	SB	I	SB R1 ADDRESS	MEM[ADDRESS] = R1

"||" symbol indicates concatenation (0100 || 1100 = 01001100).

d) The Status Register (SREG) flags are affected by the following instructions:

• The Carry flag (C) is updated every ADD instruction.
• The Overflow flag (V) is updated every ADD and SUB instruction.
• The Negative flag (N) is updated every ADD, SUB, MUL, AND, OR, SLC, and SRC instruction.
• The Sign flag (S) is updated every ADD and SUB instruction.
• The Zero flag (Z) is updated every ADD, SUB, MUL, AND, OR, SLC, and SRC instruction.
• A flag value can only be updated by the instructions related to it.

Data Path

a) Stages: 3

• All instructions regardless of their type must pass through all 3 stages.
• **Instruction Fetch (IF):** Fetches the next instruction from the main memory using the address in the PC (Program Counter), and increments the PC.
• **Instruction Decode (ID):** Decodes the instruction and reads any operands required from the register file.
• **Execute (EX):** Executes the instruction. In fact, all ALU operations are done in this stage.
Moreover, it performs any memory access required by the current instruction. For loads, it would load an operand from the main memory, while for stores, it would store an

operand into the main memory. Finally, for instructions that have a result (a destination register), it writes this result back to the register file.

b) **Number of clock cycles:** 3 + ((n − 1) ∗ 1), where n = number of instructions

– Imagine a program with 7 instructions:
∗ 3 + (6 ∗ 1) = 9 clock cycles
– You are required to understand the pattern in the example and implement it.

Package 2 Pipeline			
	Instruction Fetch (IF)	**Instruction Decode (ID)**	**Execute (EX)**
Cycle 1	Instruction 1		
Cycle 2	Instruction 2	Instruction 1	
Cycle 3	Instruction 3	Instruction 2	Instruction 1
Cycle 4	Instruction 4	Instruction 3	Instruction 2
Cycle 5	Instruction 5	Instruction 4	Instruction 3
Cycle 6	Instruction 6	Instruction 5	Instruction 4
Cycle 7	Instruction 7	Instruction 6	Instruction 5
Cycle 8		Instruction 7	Instruction 6
Cycle 9			Instruction 7

Project

General Description

Deliverables

The following guidelines must be followed in all packages:

Program Flow

a) You must write your program in **assembly language** in a text file.
b) Your must read the instructions from the text file, and parse them according to their types/formats (opcode and other relevant fields).
c) You must store the parsed version of the instructions in the memory (instruction segment of main memory or instruction memory according to your package).
d) You should start the execution of your pipelined implementation by fetching the first instruction from the memory (instruction segment of main memory or instruction memory) at Clock Cycle 1.
e) You should continue the execution based on the example provided in the Datapath section of each package reflecting the different stages working in parallel.
f) The Clock Cycles can be simulated as a variable that is incremented after finishing the required stages at a given time.
Examples:

```
Fetch();
Decode();
Execute;
// memory();
// writeback();

Cycle++
```

Printings

The following items must be printed in the console after each Clock Cycle:

a) The Clock Cycle number.
b) The Pipeline stages:
   - Which instruction is being executed at each stage?
   - What are the input parameters/values for each stage?
a) The updates occurring to the registers in case a register value was changed.
b) The updates occurring in the memory (data segment of main memory or data memory according to your package) in case a value was stored or updated in the memory.

c) The content of all registers after the last clock cycle.
d) The full content of the memory (main memory or instruction and data memories according to your package) after the last clock cycle.

Package Selection

- You are requested to submit your selection of one of the previously described packages through the link: https://docs.google.com/forms/d/e/1FAIpQLSdIA9pE1yQLsBU7aLz1l-PYCFFP4Rnb9x8UkURh9XqHf3v85Q/viewform?usp=sf_link
- **Package Selection Deadline: Saturday 06 April 2024 at 11:59 pm**
  Note: Kindly note that the final assignment will be based on the first come first serve basis to ensure equal distribution of all packages among the teams
- You can check your assigned package through the link: https://docs.google.com/spreadsheets/d/1YYNIWAF_e041cuAKfjcy7hnPAgW90Bgdipg_zeUq7RM/edit?usp=sharing

Project Instructions

Please read the following instructions carefully:

a) Any case of plagiarism will result in a zero.
b) Any case of cheating will result in a zero.
c) A cheating detection tool will be used to compare the submitted projects against all online and offline implementations similar to the project idea.
   - The projects that have more than 50% similarity percentage will receive a zero.
d) It is your responsibility to ensure that you have:
   - Submitted before the deadline.
   - Submitted the correct file(s).
   - Submitted the correct file(s) names.

Submission Guidelines

- The submission deadline for submission is **Thursday 16 May 2024 at 11:59 PM**
- You are requested to submit the following documents: The below deliverables are the ones that will be described in the deliverables section, and below is just examples of the deliverables and the naming convention (video and report is a MUST, we can add extra deliverables which are codes etc)

Project

General Description

1. A 1-min video to demonstrate the working code (please narrate and comment on the results)
   → **name the Video (Project_Team_m_Video.mp4)**
2. The required project description report (kindly include in the cover page the team number, team name, package number and name, and team members' names, IDs, and tutorials).
   → **name the report (Project_Team_m_Report.pdf)**
3. The developed C code of the experiment, the CMakeLists.txt, and any additional library used in a single zip folder
   → **name the Code (Project _Team_m_Code.zip)**

- Please upload your milestone deliverables to your drive as a .zip file with the following naming format:
  (Ex.: CSEN601_S24_Proj_Package_**n**_Team_**m**.zip)
  *where **m** is your team number and **n** is your package number.*

- Submit **ONLY** the sharing link through the below form and ***Make sure that you give permission to access***
  - https://docs.google.com/forms/d/e/1FAIpQLSf8i13CHO35cdfx-W6C-JG5l2Q450BAvaus2hPrHPZS3vYk5g/viewform?usp=sf_link

Evaluation Process

- The evaluation process of the process will be conducted during the first 2-3 days of the revision week. This timing is tentative, exact timings will be announced before the evaluations.
- The evaluation timetable will be posted on the CMS during the last teaching week.

Project Grading

The project will be graded upon multiple criteria for each of the submitted deliverables. These criteria including *(below is just some grading items and more are considered):*

- The overall functionality of the project.
- Each technical aspect of the project will be graded as well.
- The quality of the submission (for example: well-commented and generic code, comprehensive and well-written reports, clear and comprehensive videos, and others).
- Submission on time with no delays (late submissions will be subject to deduction).

Project

General Description

- The evaluation attendance is obligatory for all members and graded upon only the showing up.
- There is a collective team grade, yet during the evaluation and based on your discussion and answers, individual grades will be added as well.
- Note extra bonus marks will be added in case of successfully merging the computer systems course project and the operating systems course project (maximum bonus is 1.75% to be added to the total 100% course grade)