SOUTHERN
UNIVERSITY COLLEGE
南方大學學院

CSIS2083
Computer Organization and Architecture

Name : **LEE CHIN YONG**

Student ID : **D130447C**

Batch : **IT13-C**

Lecture : **Dr.Lee Huah**

Submit Date : **4th Dec 2015**

85

Department of Computer Science
Faculty of Engineering and Information Technology

NO:..... DATE:.....

DB0447C LEE CHIN YONG ITB-C

CPU Organisation Operation

The Fetch-Execute Cycle

The operation of the CPU[1] is usually described in terms of the Fetch-Execute cycle[2].

Fetch-Execute Cycle	The cycle raises many interesting questions, e.g.
Fetch the Instruction	An intruction is an order given to a computer processor by a computer program. At the lowest level, each instruction is a sequence of 0s and 1s that decribes a physical operation the computer is to perform and depending on the particular instruction type, the specification of special storage areas called register that may contain the instruction to be carrying out or fetch.
Increment the Program Counter	Increment the program is execute by the program counter. Program counter is a register in a computer processor that contains the address of instruction being executed at the current time. Program counter increment is stored value by 1 after the instruction fetched. The program counter points to the next instruction in the sequence.
Decode the Instruction	In order to figure out what the Instruction should do, it needs to be decorded. Part of the decoding procose fetches the input operands.
Fetch the operands	In computer, an operand is the part of a computer instruction that specifies data that is to be operating on or manipulated and by extansion, the data itself. It is the different fetching from step 1 above, the instruction describes an operation and the operand on which the operation to be performed.
Perform the operations	It is the main step of whole fetch cycle, and cannot be done simply. The operation is an elemantary operations that a computer is designed and built to perform.
Store the result	At this point, the result of the operation will written back into the register.
Repeat forever	Repeat the instruction operation, repeat for execute the new instruction operation and when the operation have error occurs, it might be cause the infinite loop.

NO:..... DATE:.....

Representing Programs

Each complex task carried out by a computer needs to broken down into a sequence of simpler tasks and a binary machine instruction is needed for the most primitive tasks. Consider a task that adds one number from second number then subtracts by the third number[2], held in memory locations designated by B, C and D[3][4] and stores the result in memory location designated by A.

A = (B + C) - D

[1] Central Processing Unit
[2] Sometimes called the Fetch-Decode-Execute Cycle
[3] Let's assume they are held in two's complement form.
[4] A, B, C and D are actually main memory addresses. i.e. natural binary numbers.

The assignment can be broken down (compiled) into a sequence of simple tasks or assembly instructions, e.g.

Assembly Instruction		Effect
LOAD	R2, B	Copy the contents of memory location designated by B into Register 2
ADD	R2, C	Add the contents of the memory location designated by C to the contents of Register 2 and put the result back into Register 2
SUB	R2, D	Subtract the contents of the memory location designated by D to the contents of Register 2 and put the result back into Register 2
STORE	R2, A	Copy the contents of Register 2 into the memory location designated by A.

Each of these assembly instructions need to be encoded into binary for execution by the Central Processing Unit (CPU). Let's try this encoding for MAS1, a simple architecture.

MAS1 Architecture.

MAS1 is a fictious architecture with the following characteristics:

1024 x 16 bit words of RAM maximum. RAM is word-addressable.

4 general purpose registers R0, R1, R2 and R3. Each general purpose register is 16-bits (the same size as a memory location).

16 different instructions that the CPU can decode and execute, e.g. LOAD, STORE, ADD, SUB and so on. These different instructions constitute the Instruction Set of the Architecture.

The representation for integers will be two's complement

For this architecture, the computer architect needs to define a coding scheme[5] for instructions. This is termed the Instruction Format.

[5]

NO:..... DATE:.....

MAS1 Instruction Format.

MAS1 Instruction are 16-bits, to fit into a main-memory word. Each instruction is divided into a number of instruction fields that encode a different piece of information for the CPU.

Field Name	OPCODE	REG	ADDRESS
Field width	4-bits	2-bits	10-bits

The OPCODE[6] field identifies the CPU operations. Since MAS1 supports 16 instructions, these can be encoded as 4-bit natural number. for MAS1, opcodes 1 to 4 will be[7]:

0001 = LOAD 0010 = STORE 0011 = ADD 0100 = SUB

[5] Most architectures actually have different instruction formats for different categories of instructions

[6] Operation Code

[7] The meaning of CPU operations is defined in the Architecture's Instruction Set Manual.

The REG field defines a General CPU Register. Arithmetic operations will use 1 register operand and 1 main memory operand, results will be written back to the register. Since MAS1 has 4 register; these can be encoded as a 2-bit natural number.

00 = Register 0 01 = Register 1 10 = Register 2 11 = Register 3

The ADDRESS field defines the address of a word in RAM. Since MAS1 can have up to 1024 memory locations; a memory address can be encoded as a 10-bit natural number. If we define address 200H, 201H, 202H and 203H for A, B, C and D, we can encode the example above as:

Assembly Instruction		Machine Instruction				
LOAD	R2, [201H]	0001	10	10	0000	0001
ADD	R2, [202H]	0011	10	10	0000	0010
SUB	R2, [203H]	0100	10	10	0000	0011
STORE	R2, [204H]	0010	10	10	0000	0000

Memory Placement of Program and Data

In order to execute a MAS1 program, its instructions and data needs to placed within main memory[8]. We'll place our 3-instruction program in memory starting at address 050H and we'll place the variables A, B, C and D at memory words 200H, 201H, 202H and 203H respectively. Such placement results in the following memory layout prior to program execution. For convenience, memory address and memory contents are also given in hex.

NO:..... DATE:.....

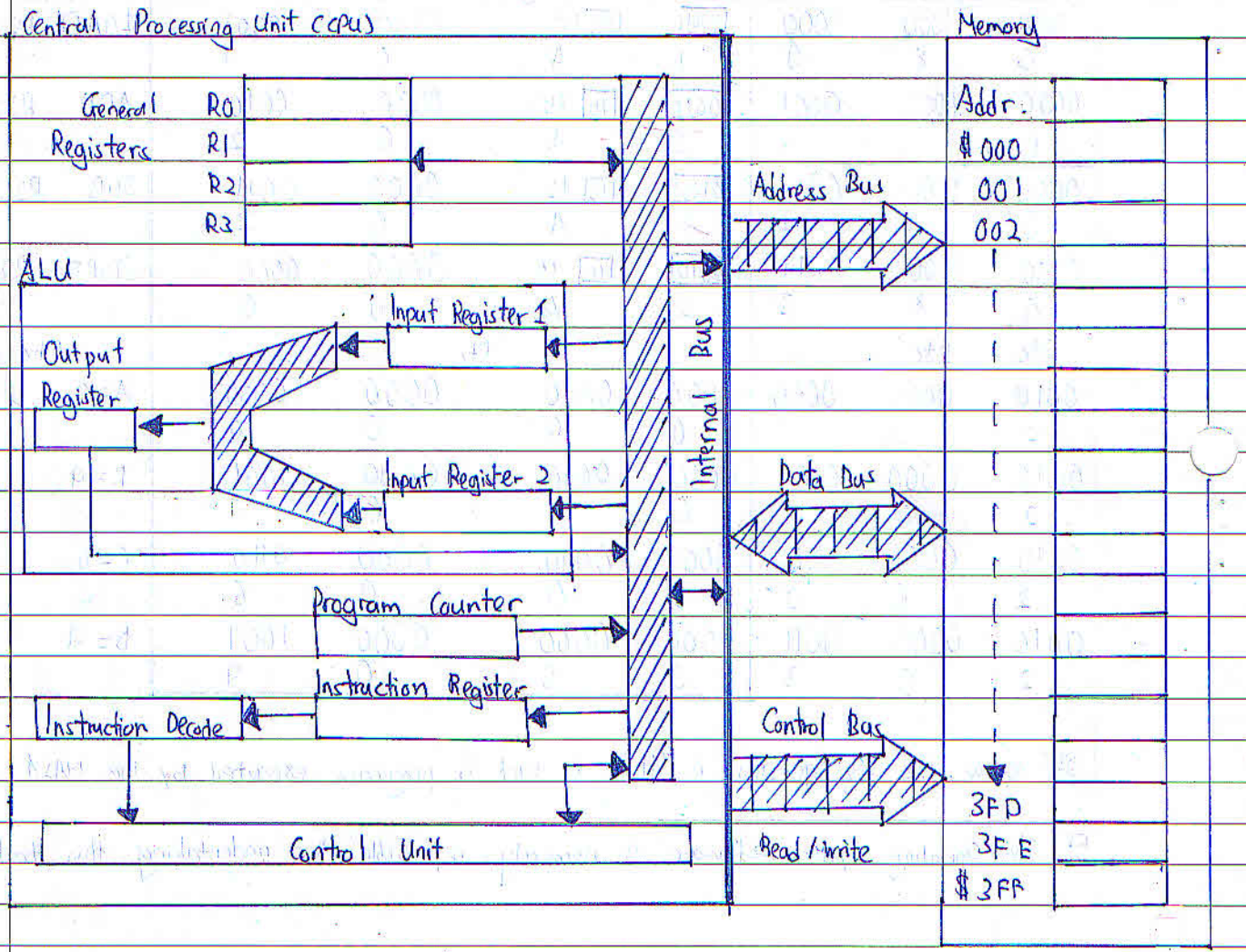
Memory Address in binary & hex			OP	Reg	Address		Assembly Instruction
0000	1000	0000	0001	10	0000	0001	LOAD R2, [201H]
0	8	0	1	A	0	1	
0000	1000	0001	0011	10	0000	0010	ADD R2, [202H]
0	8	1	3	A	0	2	
0000	1000	0010	0100	10	0000	0011	SUB R2, [203H]
0	8	2	4	A	0	3	
0000	1000	0011	0010	10	0000	0000	STORE R2, [200H]
0	8	3	2	A	0	0	
Etc			Etc				Etc
0010	0000	0000	0000	0000	0000	0000	A=0
2	0	0	0	0	0	0	
0010	0000	0001	0000	0000	0000	1001	B=9
2	0	1	0	0	0	9	
0010	0000	0010	0000	0000	0000	0110	C=6
2	0	2	0	0	0	6	
0010	0000	0011	0000	0000	0000	1001	D=9
2	0	3	0	0	0	9	

Of course, the big question is "How is such a program executed by the MAST CPU?"

CS: The Operating System software is normally responsible for undertaking this task.

NO:..... DATE:.....

CPU Organisation



The program Counter (PC) is a special register that holds the address of the next instruction to be fetched from Memory (for MAS1, the PC is 10-bits wide). The PC is incremented [by] to "point to" the next instruction while an instruction is being fetched from main memory.

The Instruction Register (IR) is a special register that holds each instruction after it is fetched from main memory. For MAS1, the IR is 16-bits since instructions are 16-bit wide.

The Instruction Decoder is a CPU component that decode and interpets the contents of the Instruction Register, i.e. it splits a whole instruction into fields for the Control Unit to interpet. The Instruction decoder is often considered to be a part of the Control Unit.

The Control Unit is the CPU component that co-ordinates all activity within the CPU. It has connections to all parts of the CPU, and includes a sophisticated timing circuit.

NO:..... DATE:.....

the Arithmetic & Logic Unit (ALU) is the CPU component that carries out arithmetic and logical operations e.g. addition, comparison, boolean AND/OR/NOT.

The ALU Input Register 1 & 2 are special registers that hold the input operands for the ALU.

The ALU Output Register is a special registers that holds the result of an ALU operation. On completion of an ALU operation the result is copied from the ALU output register to its final destination - e.g. to a CPU register, or main-memory, or to an I/O device.

[Q3] By the appropriate number of memory words.

The General-Purpose Registers R0, R1, R2, R3 are available for the programmer to use in his/her program. Typically the programmer tries to maximise the use of these registers in order to speed program execution. For MAS1, the general registers are the same size as memory location, i.e. 16-bits.

The Buses serve are communication highways for passings information within the CPU (CPU internal bus) and between the CPU and the main memory (the address bus, the data bus, and the control bus). The address bus is used to send addresses from the CPU to the main memory; these addresses indicate the memory location the CPU wishes to read or write. Unlike the address bus, the data bus is bi-directional; for writing, the data bus is used to send a word from the CPU to main-memory; for reading, the data bus is used to send a word from main-memory to the CPU. For MAS1, the Control bus [Q0] is used to indicate whether the CPU wishes to read from a memory location or write to a memory location. For simplicity we've omitted two special registers, the Memory Address Register (MAR) and the Memory Data Register (MDR). These registers lie at the boundary of the CPU and Address bus and Data bus respectively and serve to buffer data to/from the buses.

Buses can normally transfer more than 1-bit at the time. for the MAS1, the address bus is 10-bits (the size of an address), the data bus is 16-bits (size of a memory location), and the control bus is 1-bit (to indicate a memory read operation or a memory write operation).

Interlude: the Von Neumann Machine Model

Most computers conform to the von Neumann machine Model, named after the Hungarian-American mathematician John von Neumann (1903-57).

In von Neumann's model, a computer has 3 subsystems (i) a CPU (ii) a main memory and (iii) an I/O systems. The main memory holds the program as well as data and the computer is allowed to manipulate its own program [Q1]. Instructions are executed sequentially (one at a time). A single path exists between the control unit and main-memory, this leads to the so-called "von Neumann bottleneck" since memory fetches are the slowest part of an instruction they become the bottleneck in any computation.

NO:..... DATE:.....

Instruction Execution (Fetch - Execute - Cycle (Mirco - steps)

In order to execute our 3 -instruction program, the control unit has to issue and coordinate a series of micro - instructions. These micro - instructions form the fetch - execute cycle. For our example we will assume that the Program Counter Register (PC) already holds the address of the first instruction, namely 080H.

[10] Most control buses are wider than a single bit, these extras bits are used to provide more sophisticated memory operations and I/o operations.

[11] This type of manipulation is not regarded as a good technique for general assembly programming.

LOAD R2 [201H]

0000	1000	0000	0001	10	0000	0001	Copy the value in memory words 201H into Register 2
0	8	0	1	A	0	1	

Control Unit Action				Data flows
FETCH INSTRUCTION [12]				
PC to Address Bus [13]	080H	⟶	080H	Address Bus
0 to Control Bus [14]	0	⟶	0	Control Bus
Address Bus to Memory	080H	⟶	080H	Memory
Control Bus to Memory	0	READ ⟶	0	Memory
Increment PC [15]	080	INC ⟶	081H	PC becomes PC+1 [16]
Memory [080HJ to Data Bus	1A01H	⟶	1A01H	Data Bus
Data Bus to Instruction Register	1A01H	⟶	1A01H	Instruction Register
DECODE INSTRUCTION				
IR to Instruction Decoder	1A01H	⟶	1A01H	Instruction Decoder
Instruction Decoder to Control Unit [17]	1, 2, 201H	⟶	1, 2, 201H	Control Unit
EXECUTE INSTRUCTION [18]				
Control Unit to Address Bus	201H	⟶	201H	Address Bus
0 to Control Bus	0	⟶	0	Control Bus
Address Bus to Memory	201H	⟶	201H	Memory
Control Bus to Memory	0	READ ⟶	0	Memory
Memory [201HJ to Data bus	0009H	⟶	0009H	Data Bus
Data Bus to Register 2	0009H	⟶	0009H	Register 2

[12] the micro-steps is the Fetch and Decode phases are common for all instructions.

[13] This and the next 4 micro-steps initiate a fetch of the next instructions to be executed, which is to found at memory address 80+. In practice a Memory Address Register (MAR) acts as an intermediate buffer for the Address, similarly a Memory Data Register (MDR) buffers data to/from the data bus.

NO: DATE:

[14] We will use 0 for a memory READ request, and 1 for a memory WRITE request.

[15] For simplicity, we will assume that the PC is capable of performing the increment internally. If not, the Control Unit would have to transfer the contents of the PC to the ALU, get the ALU to perform the increment and send the results back to the PC. All this while we are waiting for the main-memory to return the word at address 80H

[16] Since MAS 1's main-memory is word-addressed, and all instructions are 1 word. If main-memory was byte-addressed we would need to add 2

[17] The Instruction decoder splits the instruction into individual instruction fields OPCODE, REG and ADDRESS for interpretation by the Control Unit.

[18] The micro-steps for the execute phase actually perform the operation.

ADD R2, [202H]

0000	1000	0001	0011	10	0000	0010	Add [19] the value in memory word 202H to Register 2
0	8	1	3	A	0	2	

Control Unit Action				Data flows
FETCH INSTRUCTION				
PC to Address Bus	081H	→	081H	Address Bus
0 to Control Bus	0	→	0	Control Bus
Address Bus to Memory	081H	→	081H	Memory
Control Bus to Memory	0	READ →	0	Memory
Increment PC	081H	INC →	082H	PC becomes PC+1
Memory [081H] to Data Bus	3A02H	→	3A02H	Data bus
Data Bus to Instruction Register	3A02H	→	3A02H	Instruction Register
DECODE INSTRUCTION				
IR to Instruction Decoder	3A02H	→	3A02H	Instruction Decoder
Instruction Decoder to Control Unit	3, 2, 202H	→	3, 2, 202H	Control Unit.
EXECUTE INSTRUCTION				
Register 2 to ALU Input Reg 1	0009	→	0009	ALU Input Reg 1
Control Unit to Address Bus	202H	→	202H	Address Bus
0 to Control Bus	0	→	0	Control Bus
Address Bus to Memory	202H	→	202H	Memory
Control Bus to Memory	0	READ →	0	Memory
Memory [202H] to Data bus	0006H	→	0006H	Data Bus
Data Bus to ALU Input Reg 2	0006H	→	0006H	ALU Input Reg 2
Control Unit to ALU		ADD →	000FH	Output Register
ALU Output Reg to Register 2	000F	→	000FH	Register 2

[19] Using two's complement arithmetic.

NO:..... DATE:.....

SUB R2, [203H]

0000	1000	0001	0100	10	10	0000	0011	Subtract the value of Register 2 by the value in memory word 203H
0	8	1	4	A		0	3	

Control Unit Action	Data flows			
FETCH INSTRUCTION				
PC to Address Bus	082H	→	082H	Address Bus
0 to Control Bus	0	→	0	Control Bus
Address Bus to Memory	082H	→	082H	Memory
Control Bus to Memory	0	READ →	0	Memory
Increment PC	082H	INC →	083H	PC becomes PC+1
Memory [0821H] to Data Bus	4A03H	→	4A03H	Data Bus
Data Bus to Instruction Register	4A03H	→	4A03H	Instruction Register
DECODE INSTRUCTION				
IR to Instruction Decoder	4A03H	→	4A03H	Instruction Decoder
Instruction Decoder to Control Unit	3,2,203H	→	3,2,203H	Control Unit
EXECUTE INSTRUCTION				
Register 2 to ALU Input Reg 1	000FH	→	000FH	ALU Input Reg 1
Control Unit to Address Bus	203H	→	203H	Address Bus
0 to Control Bus	0	→	0	Control Bus
Address Bus to Memory	203H	→	203H	Memory
Control Bus to Memory	0	READ →	0	Memory
Memory [202H] to Data Bus	0009H	→	0009H	Data Bus
Data Bus to ALU Input Reg 2	0009H	→	0009H	ALU Input Reg 2
Control Unit to ALU		SUB →	0006H	Output Register
ALU Output Reg to Register 2	0006H	→	0006H	Register 2

NO:..... DATE:.....

STORE R2, [200H]

0000	1000	0011	0010	10	10	0000	0000	Copy the value in Register 2 into memory word 200H
0	8	3	2	A		0	0	

Control Unit Action				Data flows
FETCH INSTRUCTION				
PC to Address Bus	083H	→	083H	Address Bus
0 to Control Bus	0	→	0	Control Bus
Address Bus to Memory	083H	→	083H	Memory
Control Bus to Memory	0	READ →	0	Memory
Increment PC	083H	INC →	084H	PC becomes PC+1
Memory [083H] to Data Bus	2A00H	→	2A00H	Data Bus
Data Bus to Instruction Register	2A00H	→	2A00H	Instruction Register
DECODE INSTRUCTION				
IR to Instruction Decoder	2A00	→	2A00	Instruction Decoder
Instruction Decoder to Control Unit	2,2,200H	→	2,2,200H	Control Unit
EXECUTE INSTRUCTION				
Register 2 to Data Bus	0006H	→	0006H	Data Bus
Control Unit to Address Bus	200H	→	200H	Address Bus
1 to Control Bus	1	→	1	Control Bus
Data Bus to Memory	0006H	→	0006H	Memory
Address Bus to Memory	200H	→	200H	Memory
Control Bus to Memory	1	WRITE →	1	Memory