SOUTHERN
UNIVERSITY COLLEGE
南方大學學院

Sem 3/ Year 2015

CSIS2083
Computer Organization and Architecture

Topic:
Assignment 2

Name	:	Lau Soon Siong
Student ID	:	D140100A
Batch	:	CS-14A
Lecture	:	Dr. Lee Huah

80

Department of Computer Science
Faculty of Engineering and Information Technology

Subject:..... No:.....

Date:

CPU Organization Operation

the fetch-Execute Cycle

The operation of the CPU is ~~usally~~ 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 instruction is an order given to a computer processor by a computer program. At the lowest level, each instruction is a sequence of os and is that decribes a physical operation the computer is to perform and depending on the partical ~~ist~~ instruction type, the specipication of special storage areas called register that may contain the instruction to be carrying out or Fetch.
Increment the Program	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 increases is stread value by 1 after the ~~is~~ instruction fetched. The program Counter point to the next instruction in the sequence
Decode the Instruction	In order to figure out what the instruction should do, it needs to be decoded. Part of the decoding process fetches the input operands.
Fetch the operands	In computer, an Operands 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 perferined.
Perform the operation	It is the main step of whole fetch cycle, and cannot be done simply. The Operation is an elemantary operation that a computer is designed and built to perform.
Store the results	At this point, the result of the operation will written basic into the register.
Repeat forever	Repeat the instruction operation, repeat for execute the new instruction operation and when the operation have error occurs, it minght be cause the infinite loop

(17)

Subject: No:

Date:

Representing Programs

Each complex task carried out by a computer needs to be broken down into a sequence of simpler tasks and a binary machine instruction is needed for the most primitive tasks. Consider a task that subtracts one number from second number then multiply by the third member[3], held in memory locations designated by B, C and D[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 address, i.e. natural binary numbers.

This assignment can be broken down (Compiled) into a squence of simpler tasks or assembly instruction, e.g.

Assembly Instruction	Effect
LOAD R2, B	Copy the content of memory location designated by B into Register 2.
ADD R2, C	ADD the content of the memory location designated by C from the content of Register 2 and put the result back into Register 2
Multiply R2, D	Multiply the content of Register 2 by the content of the memory location designated by D and put the result back into Register 2.
STORE R2, A	Copy the content of Register 2 into the memory location designated by A.

Each of these assembly instruction need to be encoded into binary for execution by the Central Processing Unit (CPU). Let's try this coding for TA, A simple architecture.

Subject: No: Date:

TA Architecture

TA is a fictitious architecture with the following characteristics:

[0 2^11 x 16-bit words of RAM maximum. RAM is word-addressable.

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

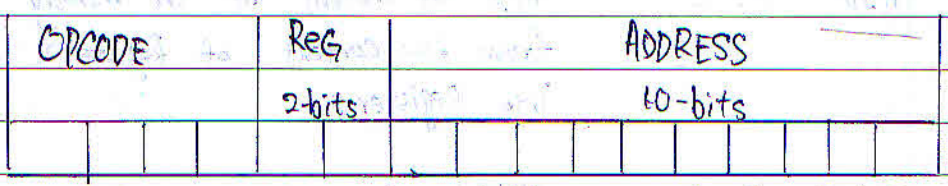
16 different instruction that the CPU can decode and execute, eg. LOAD, STORE, ADD, MULTIPLY and so on. These different instructions constitute the Instruction Set of the Architecture.

The representation for integers will be two's complement.

For this architecture, the computer architect needs to define a ~~code~~ Coding Scheme[5] for instructions. This is termed the Instruction Format.

TA Instruction Format

TA 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.



The OPCODE[6] field identifies the CPU operation. Since TA supports 16 instruction, these can be encoded as a 4-bit natural number. For TA, opcodes 1 to 5 will be[7]:

0001 = LOAD 0010 = STORE 0011 = ADD 0101 = MULTIPLY

5 Most architectures actually have different instruction formats for different categories of instruction.

6. Operation Code

7. The meaning of CPU operations is defind in the Architeture's Instruction Set Manual.

Subject: No: Date:

The REG field defines a General CPU Register. Arithmetic operation will use 1 register operand and 1 main memory operand, result will be written back to the register. Since TA has 4 register; these can be encoded as a 2-bits 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 TA 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, ~~an~~ 202H and 203H for A, B, C, 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
MULTIPLY	R2, [203H]	0101	10 10 0000 0011
STORE	R2, [200H]	0010	10 10 0000 0000

Memory Placement of Program and Data

In order to execute a TA program, its instructions and data needs to placed within main memory. We'll place our 4-instruction program in memory starting at address 080H and we'll place the variables A, B, C and D at memory words 200H, 201H, 202H, 203H respectively. Such placement result in the following memory layout prior to program execution. For convenience, memory addresses and content are also given in ~~hex~~ hex.

Subject: No: Date:

Memory Address in binary & hex	Machine Instruction OP	Reg	Address	Assembly Instruction
0000 1000 0000 0 8 0	0001	10	0000 0001 1 A 0 1	LOAD R2, [201H]
0000 1000 0001 0 8 1	0011	10	0000 0010 4 A 0 2	ADD R2, [202H]
0000 1000 0010 0 8 2	0101	10	0000 0011 3 A 0 3	MULTIPLY R2, [203H]
0000 1000 0011 0 8 3	0010	10	0000 0000 2 A 0 0	STORE R2, [200H]
Etc			Etc	Etc
0010 0000 0000 2 0 0	0000	0000	0000 0000 0 0 0 0	A=0
0010 0000 0001 2 0 1	0000	0000	0000 0010 0 0 0 2	B=2
0010 0000 0010 2 0 2	0000	0000	0000 0011 0 0 0 3	C=3
0010 0000 0011 2 0 3	0000	0000	0000 0010 0 8 0 2	D=2

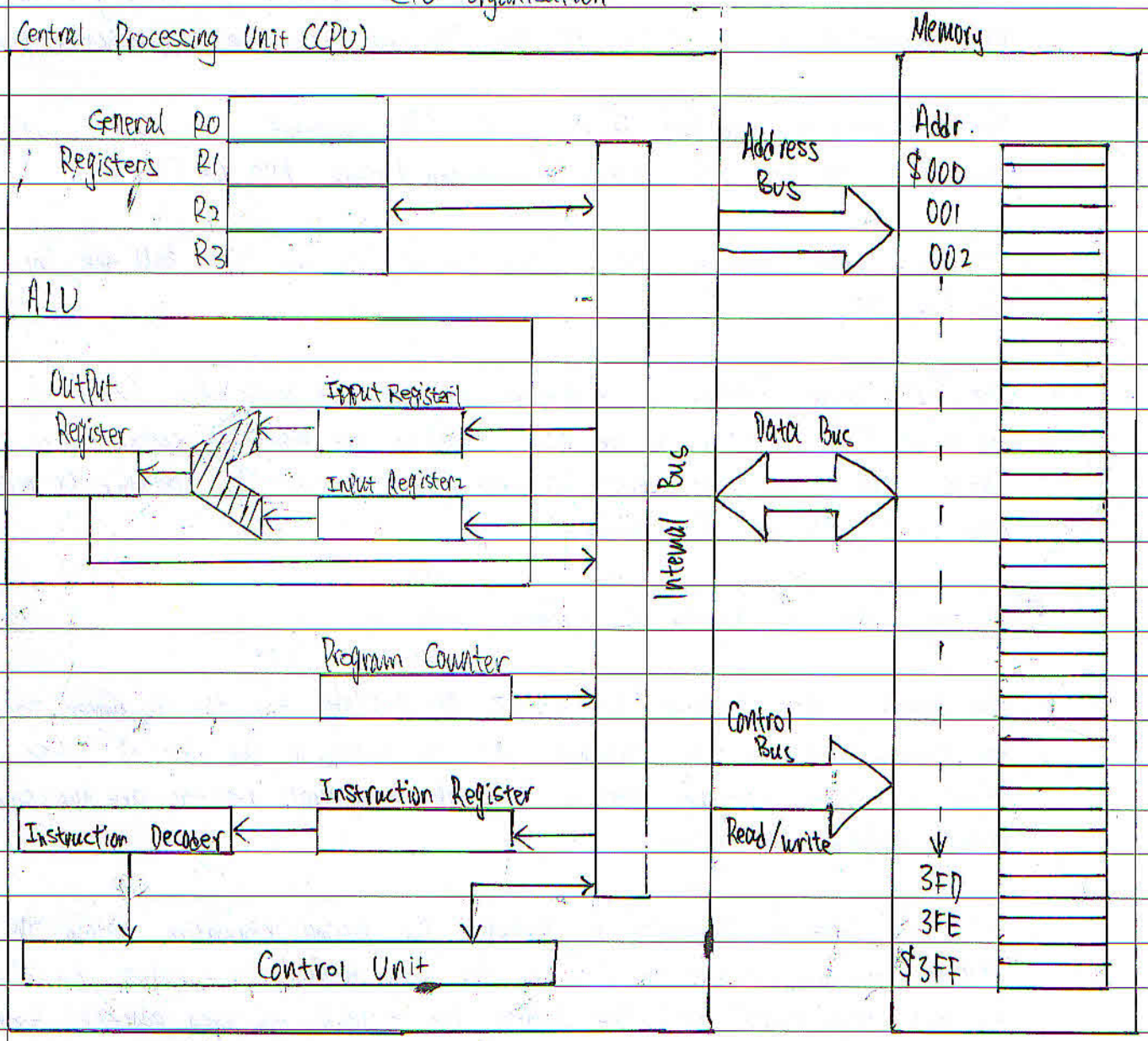
Of course, the big question is "How is such a program executed by the TA CPU?"

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

Subject:..... 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 TR, the PC is 10-bit wide). the PC is incremented 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 TR, the IR is 16-bits since instruction are 16-bits wide.

The Instruction Decoder is a CPU component that decodes and interprets the contents of the Istruction Rigister, i.e. it split a whole instruction into fields for the Control Unit to interpt. the instruction decoder is often concidered to be a part of Control Unit.

(6)

Subject: No: Date:

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

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

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

The ALU Output Register is a special register 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.

a. By the appropriate number of memory words.

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

The Buses serve as communication highways for passing 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 TA, the control bus 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.

Subject: No: Date:

Buses can normally transfer more than 1-bit at a time. For the TR, 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-bits (to indicate a memory read operation or a memory write operation).

Interlude: the Von Neumann Machine Model.

Most computers confrom to the von Neumann machine model, named after the Hungarain-American mathematician John von Neumann (1903-57).

In von Naumann's model, a computer has 3 subsystems (i) a CPU, (ii) a main memory, and (iii) an I/O system. the main memory holds the program as well as data and the computer is allowed to manipulate its own program[11]. Instruction are executed sequentially (one at a time). A single path exists between the control until and main-memory, this leads to the so-called "Von Nuemann bottleneck" since memory fetches are the slowest part of an Instruction they become the bottleneck in any computation.

Instruction Execution (Fetch-Execute-Cycle. Micro-steps)

In order to execute our 4-instruction program, the control unit has to issue and coordinate a series of micro-instruction. These micro-instruction from 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 0801H

10 Most control-buses are wider than a single bit, these extras bit are used to provide more sophisticated memory ~~operations~~ operation and I/O operations.

11 This type of manipulations is not regarded as good ~~tenig~~ technique for general assembly programming.

Subject: No: Date:

LOAD R2, [201 H]

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

Control Unit Action	Data flows			
FETCH INSTRUCTION[12]				
PC to Address Bus[13]	080 H	→	080 H	Address Bus
0 to Control Bus[14]	0	→	0	Control Bus
Address Bus to Memory	080 H	→	080 H	Memory
Control Bus to Memory	0	READ →	0	Memory
Increment PC[15]	080 H	INC →	081 H	PC become PC+1[16]
Memory [080H] to Data Bus	1A01 H	→	1A01 H	Data Bus
Data Bus to Instruction Register	1A01 H	→	1A01 H	Instruction Register
DECODE INSTRUCTION				
IR to Instruction Decoder	1A01H	→	1A01 H	Instruction Decoder
Instruction Decoder to Control Unit[17]	1, 2, 201H	→	1, 2, 201H	Control Unit
EXECUTE INSTRUCTION[18]				
Control Unit to Address Bus	201 H	→	201 H	Address Bus
0 to Control Bus	0	→	0	Control Bus
Address Bus to Memory	201 H	→	201 H	Memory
Control Bus to Memory	0	→	0	Memory
Memory [201 H] to Dat Bus	0002H	→	0002H	Data Bus
Data Bus to Register 2	0002H	→	0002H	Register 2

12 The micro-steps in the Fetch and Decode phases are common for all instruction.

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

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 result back to the PC. All this

(7)

Subject: No:

Date:

while we are waiting for the main-memory to return the word at address 80H

(6) Since TA's main-memory is word-addressed, and all instructions are 1 word. If main memory was byte-address we would need to add 2

(7) The Instruction decoder splits the instruction into the individual instruction fields OPCODE, REG and ADDRESS for interpretation by the Control Unit.

(8) The micro-steps for the execute phase actually perform the operation

ADD R2, [202H]

0000	1000	0001	0011	10	10	0000	1000	ADD the value into memory
0	8	1	3	A	0	2		word 202H from Register 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 Reg1	0002H	→	0002H	ALU Input Reg1
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	0003H	→	0003H	Data Bus
Data Bus to ALU Input Reg2	0003H	→	0003H	ALU Input Reg 2
Control Unit to ALU		ADD →	0005H	Output Register
ALU output Reg to Register2	0005H	→	0005H	Register 2

(10)

Subject: No: Date:

19. Using two's Complement arithmetic.

Multiply R2, [203 H]

0000	1000	0010	0101	10 10	0000	0011	Multiply the value of Register 2
0	8	2	5	A	0	3	by the value in memory word 203H

Control Unit Action		Data flows		
FETCH INSTRUCTION				
PC to Address Bus	0821H	→	082H	Address Bus
0 to Control Bus	0	→	0	Control Bus
Address Bus to Memory	0821H	→	0821H	Memory
Control Bus to Memory	0	→	0	Memory
Increment PC	0821H	→	0831H	PC becomes PC + 1
Memory [0821H] to Data Bus	5A03H	→	5A03H	Data Bus
Data Bus to Instruction Register	5A03H	→	5A03H	Instruction Registers.
Decode INSTRUCTION				
IR to Instruction Decoder	5A03H	→	5A03H	Instruction Decoder
Instruction Decoder to Control Unit.	5, 2, 203H	→	5, 2, 203H	Control Unit.
Execute INSTRUCTION				
Register 2 to ALU Input Reg 1	0005H	→	0005H	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	→ READ	203H	Memory
Control Bus to Memory	0	→	0	Memory
Memory [203H] to Data Bus	0002H	→	0002H	Data Bus
Data Bus to ALU Input Reg 2	0002H	→ MUL	0002H	ALU Input Reg 2
Control Unit to ALU		→	0010H	Output Register
ALU output Reg to Register 2	0010H	→	0010H	Register 2

20 Using two's complement arithmetic.

Subject: No: Date:

STORE R2, [200H]

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

Control Unit Action			Data flows	
FETCH INSTRUCTION				
PC to Address Bus	0831H	→	083H	Address Bus
0 to Control Bus	0	→	0	Control Bus
Address Bus to Memory	083H	→	083H	Memory
Control Bus to Memory	0	READ line →	0	Memory
Increment PC	083H	→	084H	PC becomes PC+1
Memory [083] 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	0010H	→	0010H	Data Bus
Control Unit to Address Bus	200H	→	200H	Address Bus
1 to Control Bus	1	→	1	Control Bus
Data Bus to Memory	0010H	→	0010H	Memory
Address Bus to Memory	200H	→	200H	memory
Control Bus to Memory	1	→	1	memory