



COURSE: CEG2136/CEG2536
Computer Architecture I
Architecture des ordinateurs I
SEMESTER: Fall 2008

FINAL EXAMINATION

Q1.1 For 1 K memory locations you need:

- (a) 8 address lines
- (b) 10 address lines
- (c) 12 address lines
- (d) None of the above

Q1.2 Which CPU register provides the address from which the next instruction opcode is to be fetched?

- (a) Instruction register IR
- (b) Accumulator AC
- (c) Program counter PC
- (d) None of these

Q1.3 What is the difference between a direct and an indirect address instruction?

How many references to memory are needed for each type of instruction to bring an operand into a processor register, given that the instruction was already fetched into IR?

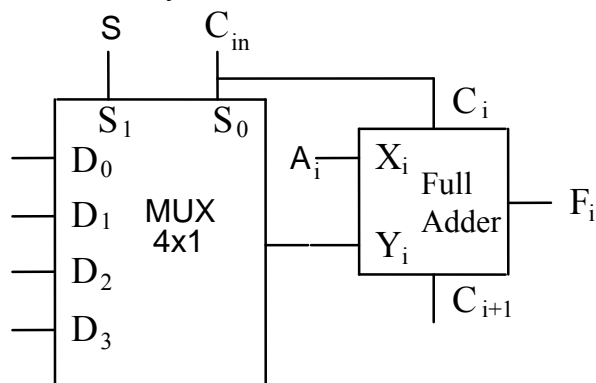
direct addressing 1
indirect addressing 2

Q1.4 An arithmetic unit performs basic operations on two numbers A and B, under the control of two bits S and C_{in} , as shown in the following table:

S	$C_{in} = 0$	$C_{in} = 1$
0	$F = A + B$ (addition)	$F = A + 1$ (increment A)
1	$F = A - 1$ (decrement A)	$F = A + B' + 1$ (subtraction)

The following combinational circuit (multiplexor and full adder) is used to implement the functions described above for bit i .

Select from the following combinations of multiplexor inputs which set of logic values implements correctly the above functions



- (a) $D_0=B_i', D_1=0, D_2=1, D_3=B_i$
- (b) $D_0=0, D_1=B_i, D_2=B_i', D_3=1$
- (c) $D_0=1, D_1=B_i', D_2=B_i, D_3=0$
- (d) $D_0=B_i, D_1=0, D_2=1, D_3=B_i'$
- (e) $D_0=1, D_1=B_i, D_2=B_i', D_3=0$

Q1.5.

The datapath of a circuit for the multiplication of two n -bit unsigned numbers is formed of:

- An n -bit register P ($P_{n-1} \dots P_0$), initially set at 0.
- Two n -bit registers for multiplier Y ($Y_{n-1} \dots Y_0$) and multiplicand X ($X_{n-1} \dots X_0$)
- A 1-bit register C for carry, initially set to 0.
- An index initially set to n .

An algorithm for the multiplication of the two n -bit unsigned numbers is defined as follows:

While Index $\neq 0$

- If $Y_0 = 1$ then $P = P + X$
- Logic right shift of $[C \parallel P \parallel Y]$.
- Index = Index - 1

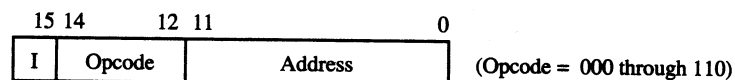
Continue (While)

The result is provided in $[P \parallel Y]$.

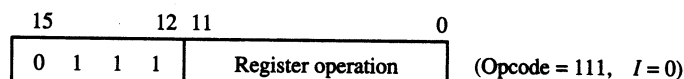
Derive the ASM chart which describes this algorithm.

Q2

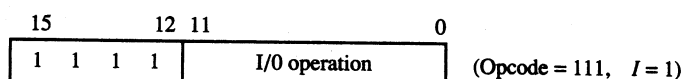
The block diagram of the basic computer that was introduced in chapter 5 of Mano's textbook is given in the annex, along with its instruction list. The instruction word is 16 bit long and has the following structure ...



(a) Memory – reference instruction



(b) Register – reference instruction



(c) Input – output instruction

- Q2.1 At some point, the content of **PC** of the basic computer is **3AF** (all numbers are in hexadecimal) and the content of **AC** is **2EC3**, as shown in the following table. The content of memory is partially given below, as well:

MEMORY		BASIC COMPUTER REGISTERS		
Address	Memory content		Content before instruction execution	Content after instruction execution
3AD	03B5	PC	3AF	3B0
3AE	ABBA	AC	2EC3	6A62
3AF	93AD	AR	0000	3B5
3B0	DEED	DR	0000	3B9F
3B1	7BEE	IR	0000	93AD
3B2	AD08	E	0	0
3B3	10BC	I	0	1
3B4	1CAA	SC		000
3B5	3B9F			
3B6	3BA0			

- What is the instruction that will be fetched and executed?
ADD 3AD I
- Show the operands and the binary operation that will be performed in the **AC** when the instruction is executed.
2EC3 + 3B9F
- Fill out the last column (“Content after instruction execution”) of the above table with the contents of registers **PC**, **AR**, **DR**, **AC**, and **IR** in hexadecimal and the values of **E**, **I** and the sequence counter **SC** in binary, all shown at the end of the instruction cycle.

- Q2.2 What is the result, in decimal, of the operation performed by the following assembly program?

ORG 100	
LDA OP	AC=FFA5
CMA	AC=005A
INC	AC=005B
ADD OP1	AC=0094
STA OP2	M(OP2)=0094
HLT	
OP1, 0039	
OP, FFA5	
OP2, 0	0094
END	

$$\text{HEX94} = \text{DEC}(9 \times 16 + 4) = 144 + 4 = 148$$

The machine code of this program is stored in a memory of 1 kilo word of 16 bits implemented on an Altera FPGA; give the Quartus .mif file that describes this program

Addr	Memory content	
		ORG 100
100	2107	LDA OP
101	7200	CMA
102	7020	INC
103	1106	ADD OP1
104	3108	STA OP2
105	7001	HLT
106	0039	OP1, 0039
107	FFA5	OP, FFA5
108	0000	OP2, 0
		END

Quartus .mif file:

Addr	+0	+1	+2	+3	+4	+5	+6	+7
100	2107	7200	7020	1106	3108	7001	0039	FFA5
108	0000	0000	0000	0000	0000	0000	0000	0000
110	0000	0000	0000	0000	0000	0000	0000	0000

- Q3** Write a subroutine to subtract two numbers that are stored in memory at 2 consecutive addresses. The address of the minuend is passed to the subroutine through the accumulator AC, while the resulted difference is passed back to the calling program through AC, as well.
- Q3.1. Write the subroutine in assembly language using the instruction list of the basic computer. The starting address of the subroutine is HEX 100.
- Q3.2. Write an example in assembly language for a main program that calls this subroutine. This program is stored in the memory starting with address 0.

```

ORG 100
DIF 0000 / return address goes here
STA MIN /save address of minuend at MIN
INC /calculate address of subtrahend
STA SUB /save address of subtrahend at SUB
LDA SUB I /load subtrahend in AC
CMA /find the 2's
INC / complement
ADD MIN I /add the minuend to the 2's complement of subtrahend
BUN DIF I /return to the calling program
MIN, 0000 /address of minuend is stored here by DIF
SUB, 0000 / address of subtrahend is stored here by DIF

```

```

ORG 0
MAIN LDA PTM /load address of the minuend into AC to pass it to the subroutine DIF
BSA DIF /call DIF
STA RES /save the result to RES
HLT
OP1 4321 /minuend
OP2 1234 /subtrahend
RES 0000 / the result will be saved here
PTM 0004 /address of minuend
PTS 0005 /address of subtrahend

```

Q4.1 Examining the annexed block diagram of the Mano's basic computer, give the list of the blocks of the datapath and the list of those that form the computer's control unit.

DATAPATH:

CONTROL UNIT:

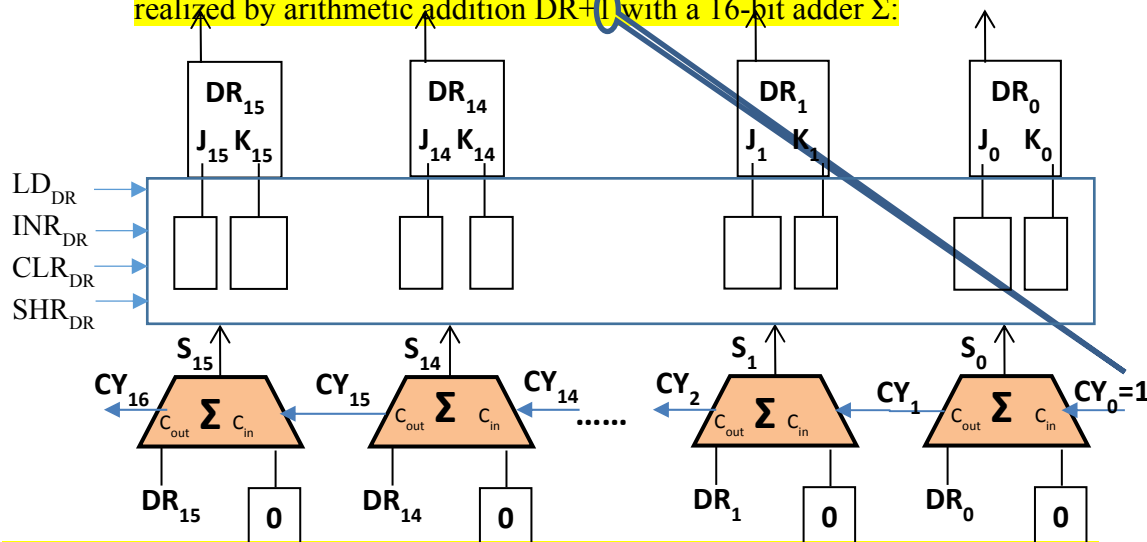
Q4.2 You have to expand the instruction list of your textbook basic computer with three more instructions (ASR, DIV, TAT) as described in the last rows of the annexed instruction list (see the ANNEX on page 7); both circuits of the datapath and the control unit need to be extended. This basic computer is a simplified version of the one presented in the textbook as it does not have any provisions to deal with interrupts.

The datapath has to be developed to allow your DR register to support the following operations:

- $LD_{DR} = x T_4$: $DR \leftarrow (bus)$ /Transfer value from bus to DR
- $INR_{DR} = y T_3$: $DR \leftarrow DR+1$ /Increment DR
- $CLR_{DR} = z T_3$: $DR \leftarrow 0$ /Clear DR
- $SHR_{DR} = v T_3 + w T_4$: $DR \leftarrow shr DR$ /Shift right DR

Design and draw the logic diagram of DR_i (bit i of register DR), using any type of combinational circuits and a JK flip-flop.

Solution: DR will be implemented as a multi-function register where incrementation INR_{DR} is realized by arithmetic addition $DR+1$ with a 16-bit adder Σ :



For arithmetic shift right DR15 has to be copied to both most significant two bits, so, for $i = 0 \dots 14$:

Function	Present state	Next state	JK FF inputs
RTL	DR_i	DR_i^+	J_{DR} K_{DR}
$LD_{DR} = x T_4$: $DR \leftarrow (bus)$	0	bus_i	bus_i $d(bus_i')$
	1	bus_i	$d(bus_i)$ bus_i'
$INR_{DR} = y T_3$: $DR \leftarrow DR+1$	0	S_i	S_i $d(S_i')$
	1	S_i	$D(S_i)$ (S_i)
$CLR_{DR} = z T_3$: $DR \leftarrow 0$	0	0	0 $d(1)$
	1	0	$D(0)$ 1
$SHR_{DR} = v T_3 + w T_4$: $DR \leftarrow shr DR$	0	DR_{i+1}	DR_{i+1} $d(DR'_{i+1})$
	1	DR_{i+1}	$d(DR_{i+1})$ DR'_{i+1}

DR_i can be eliminated from J and K equations if the inputs are made the same for both present states ($DR=0$ & $DR=1$)

$$J_{DR_i} = xT_4 bus_i + yT_3 S_i + (vT_3 + wT_4) DR_{i+1}$$

$$K_{DR_i} = xT_4 bus'_i + yT_3 S'_i + (vT_3 + wT_4) DR'_{i+1} + zT_3$$

$$J_{DR_{15}} = x T_4 bus_{15} + y T_3 S_{15} + (v T_3 + w T_4) DR_{15}$$

$$K_{DR_{15}} = xT_4 bus'_{15} + y T_3 S'_{15} + (v T_3 + w T_4) DR'_{15} + zT_3$$

Since the second Σ operand is 0, the same can be realized by using half-adders instead of full-adders.

$$S_i = DR_i \oplus CY_{i-1}$$

$$CY_i = DR_i \bullet CY_{i-1}$$

Alternatively, we can use a comprehensive table, as follows:

xT_4	yT_3	zT_3	$vT_3 + wT_4$	bus_i	S_i	DR_{i+1}	DR_i	DR_i^+	J_i	K_i
LD_{DR}	INR_{DR}	CLR_{DR}	SHR_{DR}							
1	x	x	x	0	x	x	0	0	0	x
1	x	x	x	0	x	x	1	0	x	1
1	x	x	x	1	x	x	0	1	1	x
1	x	x	x	1	x	x	1	1	x	0
x	1	x	x	x	0	x	0	0	0	x
x	1	x	x	x	0	x	1	0	x	1
x	1	x	x	x	1	x	0	1	1	x
x	1	x	x	x	1	x	1	1	x	0
x	x	1	x	x	x	x	0	0	0	x
x	x	1	x	x	x	x	1	0	x	1
x	x	x	1	x	x	0	0	0	0	x
x	x	x	1	x	x	0	1	0	x	1
x	x	x	1	x	x	1	0	1	1	x
x	x	x	1	x	x	1	1	1	x	0

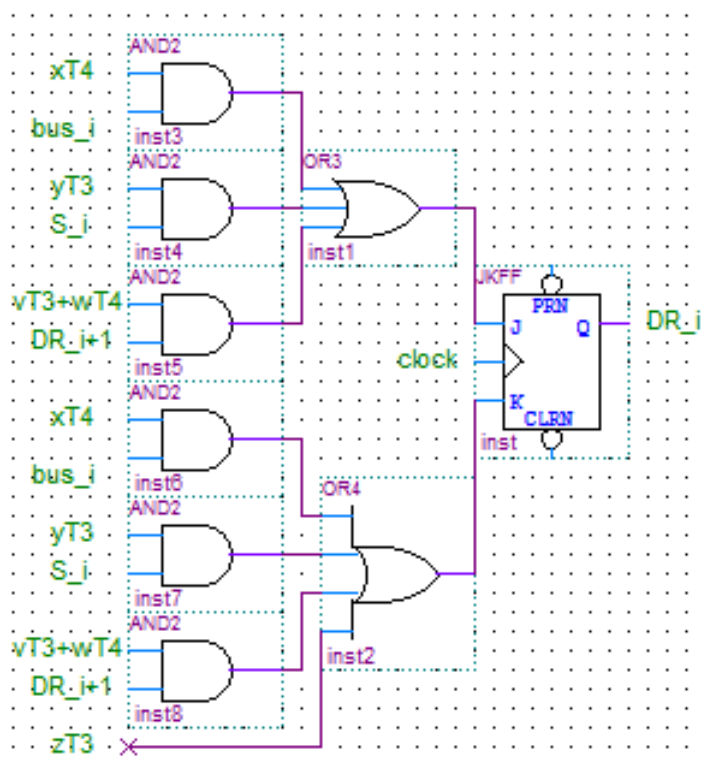
$$J_i = xT_4 bus_i + yT_3 S_i + (vT_3 + wT_4) DR_{i+1}$$

$$K_i = xT_4 bus'_i + yT_3 S'_i + zT_3 + (vT_3 + wT_4) DR'_{i+1}$$

Σ with Half-adder:

$$S_i = DR_i \oplus CY_{i-1}$$

$$CY_i = DR_i \bullet CY_{i-1}$$



Q5.1 RTL notation is used in the following table to describe the FETCH cycle of a memory reference instruction. Give short explanations of the respective microoperations in the last column

State	RTL Microoperations	Explanations
T_0	$AR \leftarrow PC$	Address of the instruction to be executed is loaded into the address register
T_1	$IR \leftarrow M[AR]$ $PC \leftarrow PC + 1$	The instruction is loaded in the Instruction Register Address of next instruction is prepared
T_2	$D_0, D_1, \dots, D_7 \leftarrow DecodeIR(14-12)$ $AR \leftarrow IR(11-0)$ $I \leftarrow IR(15)$	Opcode is decoded Address of operand (if direct addressing) or pointer to the address (if indirect addressing) goes to AR Direct/indirect bit goes to bit I
$I \cdot D_7 \cdot T_3$	$AR \leftarrow M[AR]$	

Q5.2 Use RTL notation to describe the EXECUTION cycle of each of the three newly added instructions; give the corresponding control signals in terms of the instruction code and the sequence counter. Use as many lines for your table as you need; if needed more, please use the back of the page.

ASR	F020	Arithmetic Right Shift (DR <- DR/2)
DIV	F010	Divide by 4 (DR <- DR/4)
TAT	F008	Swap AC with DR (DR <-> AC)

Solution: Use the last 3 rows of the Instruction List of the next page ANNEX to fill out the next table:

Instr.	Condition	RTL Micro-operation	Control Signals						
			Bus Select S_2, S_1, S_0	SHR _{DR}	LD _{DR}	LD _{AC}	INR _{SC}	CLR _{SC}	ALU _{DR}
ASR	$D_7IT_3IR_5$	$DR \leftarrow shr DR$ $SC \leftarrow 0$	000	1	0	0	0	1	0
TAT	$D_7IT_3IR_3$	$AC \leftarrow DR$ $DR \leftarrow AC$ $SC \leftarrow 0$	100	0	1	1	0	1	1
DIV	$D_7IT_3IR_4$	$DR \leftarrow shr DR$ $SC \leftarrow SC+1$	000	1	0	0	1	0	0
	$D_7IT_4IR_4$	$DR \leftarrow shr DR$ $SC \leftarrow 0$	000	1	0	0	0	1	0

NOTE: The term $D_7 I T_3 IR_x$ is obtained from the instruction in IR as follows:

ASR	F	0	2	0
	1111	0000	0010	0000
	I D ₇		IR ₅	
TAT	F	0	1	0
	1111	0000	0001	0000
	I D ₇		IR ₄	
DIV	F	0	0	8
	1111	0000	0000	1000
	I D ₇			IR ₃

Derive the equations of the control signals from the above table:

$$SHR_{DR} = D_7IT_3IR_5 + D_7IT_3IR_4 + D_7IT_4IR_4$$

$$LD_{DR} = D_7IT_3IR_3 = LD_{AC} = ALU_{DR}$$

$$INR_{SC} = D_7IT_3IR_4$$

$$CLR_{SC} = D_7IT_3IR_3$$

ANNEX**Instruction List of the Basic Computer**

I=0 Direct addr.		I=1 Indirect addr.		Description
Assembly language syntax	Machine Code (Hex)	Assembly language syntax	Machine Code (Hex)	
AND adr	0adr	AND adr i	8(adr)	AND memory word M to AC
ADD adr	1(adr)	ADD adr i	9(adr)	Add memory word M to AC, carry to E
LDA adr	2(adr)	LDA adr i	A(adr)	Load memory word from M to AC
STA adr	3(adr)	STA adr i	B(adr)	Store content of AC in memory M
BUN adr	4(adr)	BUN adr i	C(adr)	Branch unconditionally
BSA adr	5(adr)	BSA adr i	D(adr)	Save return Address in m, Branch to m+1
ISZ adr	6(adr)	ISZ adr i	E(adr)	Increment memory word M & skip if zero
CLA	7800			Clear AC
CLE	7400			Clear E
CMA	7200			Complement AC
CME	7100			Complement E
CIR	7080			Circulate Right E and AC
CIL	7040			Circulate Left E and AC
INC	7020			Increment AC
SPA	7010			Skip next instruction if AC is positive
SNA	7008			Skip next instruction if AC is negative
SZA	7004			Skip next instruction if AC is zero
SZE	7002			Skip next instruction if E is zero
HLT	7001			Halt computer
		INP	F800	Input character to AC and Clear Flag
		OUT	F400	Output character from AC and Clear Flag
		SKI	F200	Skip if Input Flag is on
		SKO	F100	Skip if Output Flag is on
		ION	F080	Turn Interrupt on
		IOF	F040	Turn Interrupt off
		ASR	F020	Arithmetic Right Shift (DR <- DR/2)
		DIV	F010	Divide by 4 (DR <- DR/4)
		TAT	F008	Swap AC with DR (DR <-> AC)

Basic Computer Block Diagram

