THE OPEN UNIVERSITY OF SRI LANKA
Faculty of Engineering Technology
Department of Electrical & Computer Engineering

Bachelor of Technology in Engineering Honours

Final Examination (2016/2017)
ECX5236: Computer Architecture

Date: 23rd November 2017 **Time: 9:30 pm – 12:30 pm**

---

1. Answer any five (5) questions. All questions carry equal marks.
2. Assume reasonable values or any suitable assumptions for any data not given in or if any doubt as to the interpretation of the wording of a question. Clearly state such assumptions made on the script.
3. You are allowed to use scientific calculators during the exam.
4. You are NOT allowed to use any study material or any other electronic resource during the examination.

Q1.

a) Describe major tasks of an I/O system.

[06 Marks]

b) Briefly describe Programmed I/O, Memory mapped I/O. Interrupt I/O and Co-Processor I/O with appropriate diagrams.

[06 Marks]

c) Find the efficiency of a processor that has following Programmed I/O system: Instruction execution rate – 0.15 MIPS, Number of instructions that needed by I/O subroutine to write to a disk – 10,000.

[04 Marks]

d) Assume that an Interrupt I/O is added to the processor given in question (Q1.c) and now disk has total latency of 100ms. Initiating the disk transfer and respond to the interrupt at the end of the disk latency requires 200 instructions. Find the useful time and the efficiency of the system.

[04 Marks]

Q2.

a) The cycle time t of an instruction pipeline is the time needed to advance a set of instructions one stage through the pipeline. Derive an equation for the total time required for a pipeline $T_{k,n}$, that needs k stages to execute n instructions.

[04 Marks]

b) Derive an equation for the speedup factor for the instruction pipeline compared to execution without the pipeline.

[04 Marks]

c) Consider an instruction sequence of length n that is streaming through the instruction pipeline. Let p be the probability of encountering a conditional or unconditional branch instruction, and let q be the probability that execution of a branch instruction I causes a jump to a nonconsecutive address. Assume that each such jump requires the pipeline to flushing all ongoing instruction processing, when I emerges from the last stage. Revise above derived equations (a) and (b) and derive better approximations for each by taking these probabilities into account.

[08 Marks]

d) Describe pipeline hazards and mention what are the hazards that can occur in a system like above (Q2.c).

[04 Marks]

Q3.

a) What are the cache memory mapping techniques and describe the advantages and disadvantages of each of them.

[04 Marks]

b) Consider a disk with 5400 rpm disk rotational speed, average seek time 30ms, 512-byte block size, 128-byte inter-block gap size, 20 blocks per track, 400 tracks per surface and 8 double sided disks.

i. Calculate the total capacity and useful capacity of a cylinder.

[04 Marks]

ii. If one track of data can be transferred per revolution, what is the data transfer rate

[04 Marks]

iii. Calculate the maximum rotational delay.

[04 Marks]

iv. What is the average time to locate and transfer a single sector given its address

[04 Marks]

Q4.

You are asked to implement a computer based pulse oximeter. Assume that there are sensor units, which detect oxygen saturation of the blood and send 8-bit digital data to the computer as a stream. Note that you have to use the appropriate register names provided in the Appendix A and exact values (Ex: configuring/ reading register names and values, flag names, etc.), when you are answering following questions.

a) Now a sensor unit is connected via COM1 port (RS 232 protocol). After processing data, you have to send data via COM2 port to an external display unit. Assume ports and externally connected device are based on 8250 family UART.

I. Draw the flow chart of the receiving subroutine.

[03 Marks]

II. Draw the flow chart of the transmitting subroutine.

[03 Marks]

b) Now you have to connect **two** sensor units to the same computer (one computer with two sensor units) via two ISA cards (Appendix B). You have to switch between two cards one after one to receive and then send data (one subroutine for receive form both card and another subroutine for send data to both cards).

I. Draw a block diagram to show the connectivity of the sensor unit with the computer through the ISA cards.

[03 Marks]

II. Briefly describe how you switch between ISA cards.

[02 Marks]

III. Draw the flow chart for the receiving subroutine.

[03 Marks]

IV. Write an 8086/88 assembly program for the above two subroutines (Refer Appendix C for 8086/88 ISA).

[06 Marks]

Q5.

a) Compare and contrast Superscalar and VLIW architectures.

[04 Marks]

b) Describe and distinguish Systolic and MIMD architectures.

[04 Marks]

c) You are asked to design a Co-processor for Matrix multiplication. The Co-processor has to speed up the process using special hardware architectures and techniques (Ex: Parallel processing, multi core architectures, pipeline, etc.). Mention if you make any assumptions.

I. Draw the external view of the Co-processor (Co-processor and its inputs and outputs).

[02 Marks]

II. Design and draw a memory unit block or blocks for the above Co-processor and propose a suitable indexing method for it.

[04 Marks]

III. Design and draw the block diagram of the (architecture) Co-Processor and mention the additional details of the design. You should clearly show the processing elements (PEs), memory units, data paths, etc.

[06 Marks]

Q6.

a) Compare and contrast RAID 4 and RAID 5 systems and explain the advantages and disadvantages of each.

[04 Marks]

b) In RAID systems, what is the distinction between parallel access and independent access

[04 Marks]

c) Consider machine A and machine B that have a clock rate 2Ghz, with two different instruction sets. Details of each machine are recorded Table 6.1 by running a benchmark program. Find the **MIPS rate**, and **execution time** for each machine.

Table 6.1

Machine	Instruction Type	Instructions Count (millions)	Cycles per Instruction
A	Arithmetic and logic	8	1
	Load and Store	4	3
	Branch	2	4
	Others	4	3
B	Arithmetic and logic	10	1
	Load and Store	8	2
	Branch	2	4
	Others	4	3

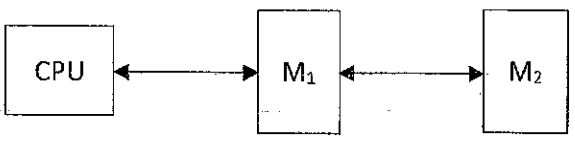
[08 Marks]

d) State another **two** ways that can be used to measure the Steady State Performance of a processor and compare (advantages and disadvantages) each method with MIPS.

[04 Marks]

Q7.

Consider two-level memory hierarchy (M_1, M_2) for a computer system, as depicted in the following diagram.



Let C_1 and C_2 be the costs per bit, S_1 and s_2 be the storage capacities, and t_1 and t_2 be the access times of the memories M_1 and M_2 respectively. The hit ratio H is defined as the probability that a logical address generated by the CPU refers to information stored in the memory M_1.

a) What is the average cost C per bit of the entire memory hierarchy?

[04 Marks]

b) Under what condition will the average cost per bit C approach C_2?

[04 Marks]

c) What is the average access time t_a for the CPU to access a word from the memory system?

[04 Marks]

d) Let $r = t_2/t_1$ be the speed ratio of the two memories. Let E=t1/ta be the access efficiency of the memory system. Express E in terms of r and H.

[04 Marks]

e) If the speed ratio of the two memories is 100, what is he required minimum value of the hit ratio to make the access efficiency of the memory system over 0.90?

[4 Marks]

Q8.

a) Comment on following statement giving reasons.
"Processing power of a processor can be increased by increasing the size of the area and number of transistors and there is no limit for that"

[04 Marks]

b) Write brief descriptions on Re-configurable architectures, Quantum computing, Hyper-Threading Technology and System-on-a-Chip (SoC).

[08 Marks]

c) What are the differences (Architectures, cost, performance, etc.) between CPU and GPU?

[04 Marks]

d) Let α be the percentage of program code that can be executed simultaneously by n processors in a computer system. Assume that the remaining code must be executed sequentially by a single processor. Each processor has an execution rate of x MIPS). Derive an expression for the effective MIPS rate when using the system for exclusive execution of this program, in terms of n, α and x.

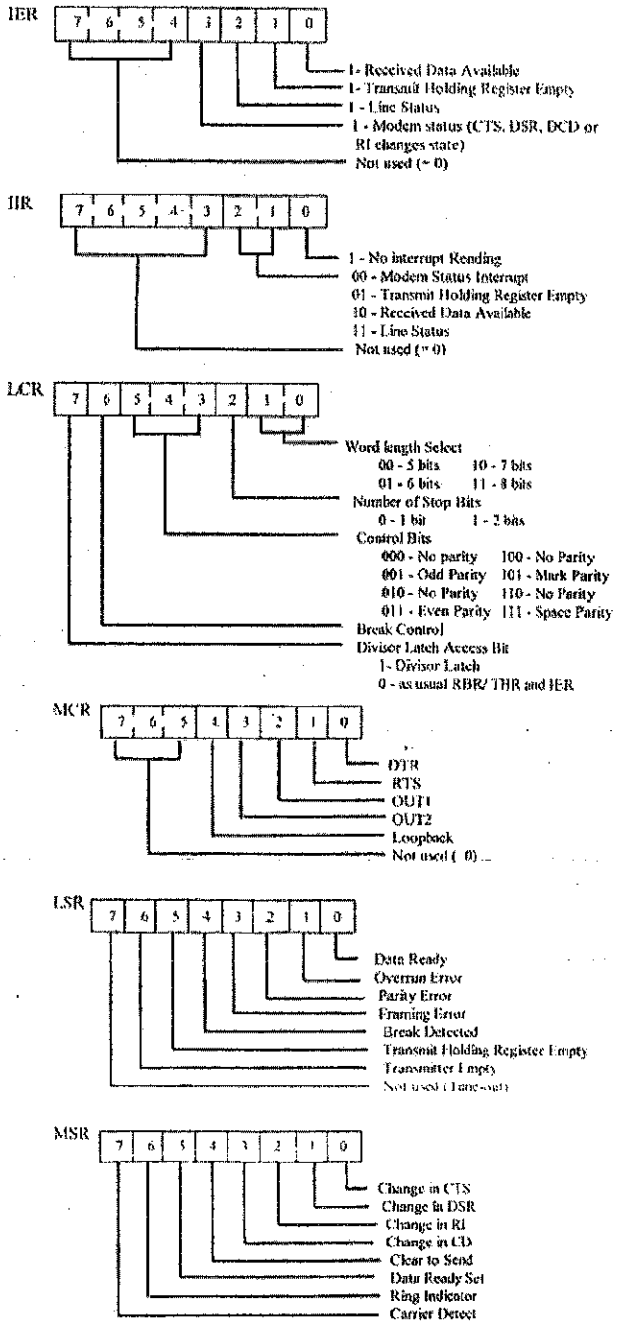
[04 Marks]

Appendix A

8250 Family Register Definitions

Register Name	Offset	Abbreviation	Access Type
Receiver Buffer Register	0	RBR	Read Only
Transmit Holding Register	0	THR	Write Only
Interrupt Enable Register	1	IER	Read/Write
Interrupt Identification Register	2	IIR	Read Only
FIFO Control Register (16550)	2	FCR	Write Only
Line Control Register	3	LCR	Read/Write
Modem Control Register	4	MCR	Read/Write
Line Status Register	5	LSR	Read Only
Modem Status Register	6	MSR	Read Only
Divisor Latch (16 bits)	0/1	DL	Read/Write

Base address of COM1 - 3F8h Base address of COM2 - 2F8h



Appendix B

ISA (Industry Standard Architecture) Bus

Pins layout

D	Description	Name	Pin	Pin	Name	Description	D
/	Ground	GND	B1	A1	IO CHK	I/O Channel Check	-I
O	Reset	RESET	B2	A2	SD7	System Data bit 7	I/O
/	Power +5V	+5V	B3	A3	SD6	System Data bit 6	I/O
I	Interrupt Request 9	IRQ9	B4	A4	SD5	System Data bit 5	I/O
/	Power -5V	-5V	B5	A5	SD4	System Data bit 4	I/O
I	DMA Request 2	DRQ2	B6	A6	SD3	System Data bit 3	I/O
/	Power -12V	-12V	B7	A7	SD2	System Data bit 2	I/O
/	Zero Wait State	0WS	B8	A8	SD1	System Data bit 1	I/O
/	Power +12V	+12V	B9	A9	SD0	System Data bit 0	I/O
/	Ground	GND	B10	A10	IO RDY	I/0 Channel Ready	-I
O	System Memory Write	SMEMW	B11	A11	AEN	Address Enable	O
O	System Memory Read	SMEMR	B12	A12	SA19	System Address bit 19	O
I/O	I/O Write	IOW	B13	A13	SA18	System Address bit 18	O
I/O	I/O Read	IOR	B14	A14	SA17	System Address bit 17	O
O	DMA Request Acknowledge 3	DACK3	B15	A15	SA16	System Address bit 16	O
I	DMA Request 3	DRQ3	B16	A16	SA15	System Address bit 15	O
O	DMA Request Acknowledge 1	DACK1	B17	A17	SA14	System Address bit 14	O
I	DMA Request 1	DRQ1	B18	A18	SA13	System Address bit 13	O
I/O	Refresh cycle in progress	REFRESH	B19	A19	SA12	System Address bit 12	O
O	System Clock	CLOCK	B20	A20	SA11	System Address bit 11	O
I	Interrupt Request 7	IRQ7	B21	A21	SA10	System Address bit 10	O
I	Interrupt Request 6	IRQ6	B22	A22	SA9	System Address bit 9	O
I	Interrupt Request 5	IRQ5	B23	A23	SA8	System Address bit 8	O
I	Interrupt Request 4	IRQ4	B24	A24	SA7	System Address bit 7	O
I	Interrupt Request 3	IRQ3	B25	A25	SA6	System Address bit 6	O
O	DMA Request Acknowledge 2	DACK2	B26	A26	SA5	System Address bit 5	O
O	T/C	TC	B27	A27	SA4	System Address bit 4	O
?	Buffered Address Latch Enable	BALE	B28	A28	SA3	System Address bit 3	O
/	Power +5V	+5V	B29	A29	SA2	System Address bit 2	O
O	Oscillator	OSC	B30	A30	SA1	System Address bit 1	O
/	Ground	GND	B31	A31	SA0	System Address bit 0	O

Signal Description

CLOCK (*System Drive*) output
The system clock is a synchronous microprocessor cycle clock.

RESET (*Reset Drive*) output
This signal goes high at power-up, hardware reset, or when low line-voltage occurs.

SA0 to SA19 (*System Addresses*) input/output
The system address lines run from bit 0 through 19. They are latched onto the falling edge of BALE.

SD0 to SD7 (*System Data bits*) Input/Output
System data bits 0 to 7.

BALE (*Buffered Address Latch Enable*) input
The buffered address latch enable is used in latch SA0 to SA19 on the falling edge of BALE. During DMA cycles, BALE is forced high.

IO CHK (*I/O Channel Check*) active low input
I/O channel check is active low signal which indicate that a parity error exists in the I/O board.

IO RDY (*I/O Channel Ready*) input
This signal lengthens I/O or memory cycles and should be held low with valid addresses. It can be held low for a maximum of 2.5 microseconds.

IRQ 3 to 7, 9 (*Interrupt Requests*) input
These interrupt request signals indicate I/O service request attention. They are prioritized in the following sequences: highest IRQ 9 and lowest IRQ 3, 4, 5,6,7,8.

IOR (*I/O Read*) active low input/output
Instructs an I/O device to drive its data onto the data bus.

IOW (*I/O Write*) active low output
Instructs an I/O device to read the data off the data bus.

SMEMR (*System Memory Read*) output
The system memory read signal is low while the low first megabyte memory is being read.

SMEMW (*System Memory Write*) output
The system memory write signal is low while the low first megabyte memory is being written.

DRQ 0 to 3 (*DMA Requests*) active high input
DMA Request channels 0 to 3 are for 8-bit data transfers. DRQ4 is used on the system board. Hold a DRQ line high until its DMA Request Acknowledge (DACK) goes active. Their priority is in the following sequences: highest DRQ 0,1,2, and 3.

DACK 1 to 3 (*DMA Request Acknowledges*) output
These signals are used to acknowledge the corresponding signals for DRQ 0 to 3.

AEN (*Address Enable*) output
The address enable is high when the DMA controller drives the address bus and is low when the CPU drives the address bus.

REFRESH (*Refresh cycle in progress*) active low input/output
This signal indicates a refresh cycle is in progress.

TC (*T/C*) output

OSC (*Oscillator*) output
The oscillator signal is used for the color graphic card.
High-speed clock (70 ns, 14.31818 MHz), 50%duty cycle

0WS (*Zero Wait State*) input
The zero wait state indicates to the microprocessor that the present bus cycle can be completed without inserting any additional wait cycles.

Appendix C

8086/88 Instruction Set Summary

Mnemonic	Instruction	Description	Operation	Flags								
				O	D	I	T	S	Z	A	P	C
MOV	MOV Dest,Source	Move (copy)	Dest:=Source									
XCHG	XCHG Op1,Op2	Exchange	Op1:=Op2 , Op2:=Op1									
STC	STC	Set Carry	CF:=1									1
CLC	CLC	Clear Carry	CF:=0									0
CMC	CMC	Complement Carry	CF:= ¬ CF									x
STD	STD	Set Direction	DF:=1 (string op's downwards)		1							
CLD	CLD	Clear Direction	DF:=0 (string op's upwards)		0							
STI	STI	Set Interrupt	IF:=1			1						
CLI	CLI	Clear Interrupt	IF:=0			0						
PUSH	PUSH Source	Push onto stack	DEC SP, [SP]:=Source									
PUSHF	PUSHF	Push flags	O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL									
PUSHA	PUSHA	Push all general registers	AX, CX, DX, BX, SP, BP, SI, DI									
POP	POP Dest	Pop from stack	Dest:=[SP], INC SP									
POPF	POPF	Pop flags	O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL	x	x	x	x	x	x	x	x	x
POPA	POPA	Pop all general registers	DI, SI, BP, SP, BX, DX, CX, AX									
CBW	CBW	Convert byte to word	AX:=AL (signed)									
CWD	CWD	Convert word to double	DX:AX:=AX (signed)	x				x	x	x	x	x
CWDE	CWDE	Conv word extended double	EAX:=AX (signed)									
IN	IN Dest, Port	Input	AL/AX/EAX := byte/word/double of specified port									
OUT	OUT Port, Source	Output	Byte/word/double of specified port := AL/AX/EAX									
ADD	ADD Dest,Source	Add	Dest:=Dest+Source	x				x	x	x	x	x
ADC	ADC Dest,Source	Add with Carry	Dest:=Dest+Source+CF	x				x	x	x	x	x
SUB	SUB Dest,Source	Subtract	Dest:=Dest-Source	x				x	x	x	x	x
SBB	SBB Dest,Source	Subtract with borrow	Dest:=Dest-(Source+CF)	x				x	x	x	x	x
DIV	DIV Op	Divide (unsigned)	Op=byte: AL:=AX / Op AH:=Rest	$				$	$	$	$	$
DIV	DIV Op	Divide (unsigned)	Op=word: AX:=DX:AX / Op DX:=Rest	$				$	$	$	$	$
DIV	DIV Op	Divide (unsigned)	Op=doublew.: EAX:=EDX:EAX / Op EDX:=Rest	$				$	$	$	$	$
IDIV	IDIV Op	Signed Integer Divide	Op=byte: AL:=AX / Op AH:=Rest	$				$	$	$	$	$
IDIV	IDIV Op	Signed Integer Divide	Op=word: AX:=DX:AX / Op DX:=Rest	$				$	$	$	$	$
IDIV	IDIV Op	Signed Integer Divide	Op=doublew.: EAX:=EDX:EAX / Op EDX:=Rest	$				$	$	$	$	$
MUL	MUL Op	Multiply (unsigned)	Op=byte: AX:=AL*Op if AH=0	x				$	$	$	$	x
MUL	MUL Op	Multiply (unsigned)	Op=word: DX:AX:=AX*Op if DX=0	x				$	$	$	$	x
MUL	MUL Op	Multiply (unsigned)	Op=double: EDX:EAX:=EAX*Op if EDX=0	x				$	$	$	$	x
IMUL	IMUL Op	Signed Integer Multiply	Op=byte: AX:=AL*Op if AL sufficient	x				$	$	$	$	x
IMUL	IMUL Op	Signed Integer Multiply	Op=word: DX:AX:=AX*Op if AX sufficient	x				$	$	$	$	x
IMUL	IMUL Op	Signed Integer Multiply	Op=double: EDX:EAX:=EAX*Op if EAX sufficient	x				$	$	$	$	x
INC	INC Op	Increment	Op:=Op+1 (Carry not affected !)	x				x	x	x	x	
DEC	DEC Op	Decrement	Op:=Op-1 (Carry not affected !)	x				x	x	x	x	

CMP	CMP Op1,Op2	Compare	Op1-Op2	x				x	x	x	x	x
SAL	SAL Op,Quantity	Shift arithmetic left (SHL)						x	x	$	x	x
SAR	SAR Op,Quantity	Shift arithmetic right						x	x	$	x	x
RCL	RCL Op,Quantity	Rotate left through Carry										x
RCR	RCR Op,Quantity	Rotate right through Carry										x
ROL	ROL Op,Quantity	Rotate left										x
ROR	ROR Op,Quantity	Rotate right										x
NEG	NEG Op	Negate (two-complement)	Op:=0-Op if Op=0 then CF:=0 else CF:=1	x				x	x	x	x	x
NOT	NOT Op	Invert each bit	Op:= Op (invert each bit)									
AND	AND Dest,Source	Logical and	Dest:=Dest ∧ Source	0				x	x	$	x	0
OR	OR Dest,Source	Logical or	Dest:=Dest∨Source	0				x	x	$	x	0
XOR	XOR Dest,Source	Logical exclusive or	Dest:=Dest (exor) Source	0				x	x	$	x	0
SHL	SHL Op,Quantity	Shift logical left(SAL)						x	x	$	x	x
SHR	SHR Op,Quantity	Shift logical right						x	x	$	x	x
NOP	NOP	No operation	No operation									
LEA	LEA Dest,Source	Load effective address	Dest := address of Source									
INT	INT Nr	Interrupt	interrupts current program, runs spec. int-program			0	0					
CALL	CALL Proc	Call subroutine										
JMP	JMP Dest	Jump										
JE	JE Dest	Jump if Equal	(JZ)									
JZ	JZ Dest	Jump if Zero	(JE)									
JCXZ	JCXZ Dest	Jump if CX Zero										
JP	JP Dest	Jump if Parity (Parity Even)	(JPE)									
JPE	JPE Dest	Jump if Parity Even	(JP)									
RET	RET	Return from subroutine										
JNE	JNE Dest	Jump if not Equal	(JNZ)									
JNZ	JNZ Dest	Jump if not Zero	(JNE)									
JECXZ	JECXZ Dest	Jump if ECX Zero										
JNP	JNP Dest	Jump if no Parity (Parity Odd)	(JPO)									
JPO	JPO Dest	Jump if Parity Odd	(JNP)									
JA	JA Dest	Jump if Above	(JNBE)									
JAE	JAE Dest	Jump if Above or Equal	(JNB / JNC)									
JB	JB Dest	Jump if Below	(JNAE / JC)									
JBE	JBE Dest	Jump if Below or Equal	(JNA)									
JNA	JNA Dest	Jump if not Above	(JBE)									
JNAE	JNAE Dest	Jump if not Above or Equal	(JB / JC)									
JNB	JNB Dest	Jump if not Below	(JAE / JNC)									
JNBE	JNBE Dest	Jump if not Below or Equal	(JA)									
JC	JC Dest	Jump if Carry										
JNC	JNC Dest	Jump if no Carry										
JG	JG Dest	Jump if Greater	(JNLE)									
JGE	JGE Dest	Jump if Greater or Equal	(JNL)									
JL	JL Dest	Jump if Less	(JNGE)									
JLE	JLE Dest	Jump if Less or Equal	(JNG)									
JNG	JNG Dest	Jump if not Greater	(JLE)									
JNGE	JNGE Dest	Jump if not Greater or Equal	(JL)									

JNL	JNL Dest	Jump if not Less	(JGE)										
JNLE	JNLE Dest	Jump if not Less or Equal	(JG)										
JO	JO Dest	Jump if Overflow											
JNO	JNO Dest	Jump if no Overflow											
JS	JS Dest	Jump if Sign (= negative)											
JNS	JNS Dest	Jump if no Sign (= positive)											

Op: Operand, Dest: Destination

Flags: x : affected by particular instruction
$: undefined after this instruction

Flags Description:

D: Direction	1 = string operation's process down from high to low address
I: Interrupt	Whether interrupts can occur. 1= enabled
T: Trap	Single step for debugging
C: Carry	Result of unsigned operation. Is too large or below zero. 1 = carry/borrow
O: Overflow	Result of signed operation. Is too large or small. 1 = overflow/underflow
S: Sign	Sign of result. Reasonable for Integer only. 1 = negative / 0 = positive
Z: Zero	Result of operation is zero. 1 = zero
A: Aux.	carry similar to Carry but restricted to the low nibble only
P: Parity	1 = result has even number of set bits