|  |  |  |
| --- | --- | --- |
| **Cairo University** | **CMP201A** | **Total:100 Points** |
| **Faculty of Engineering** | **Microprocessor Systems** | **2019-2020** |
| **Computer Eng. Department** | **Final Exam** | **Two Hour** |

**This is an open-book, open notes exam. All electronic devices - Except calculators - are forbidden.  
Make any reasonable assumptions (if necessary)   
Answer the following questions**

**Q1 – A - [12] Indicate if the statement is True or False by writing √ or X**

|  |  |
| --- | --- |
| SUB command changes the flag register the same as CMP command do | ***T*** |
| The main advantage of 8086 over 8085 is pipelining | ***T*** |
| Any assembly program must have at least two segments | ***F*** |
| In 8086, the command **POP AL** make the reverse operation of **PUSH AL** | ***F*** |
| Macros is always better than procedures for large assembly programs | ***F*** |
| IMUL to MUL is the same as JG to JA | ***T*** |

**Q1 – B - [5] What does the following abbreviations stands for:**

*Note: Abbreviations according to this course only, don’t innovate here please*

|  |  |
| --- | --- |
| **DB** | **Define Byte** |
| **RTS** | **Request To Send** |
| **DCD** | **Data Connection Detected** |
| **DUP** | **Duplicate** |
| **TF** | **Trap Flag** |

**Q1 – C - [10] What does the following programs do? [Results saved in RESULT variable]***Note: RESULT variable content is not required. It is required the relation between RESULT and other vars*

|  |  |
| --- | --- |
| .MODEL SMALL  .DATA  X DB 5  RESULT DW ?  .CODE  MAIN PROC FAR  MOV AX,@DATA  MOV DS,AX  MOV CL, X  MOV AX,1  SHL AX, CL  MOV RESULT, AX  MAIN ENDP  END MAIN | .MODEL SMALL  .DATA  X DB 2  Y DB 4  RESULT DW ?  .CODE  MAIN PROC FAR  MOV AX,@DATA  MOV DS,AX  MOV AL, X  MOV AH,Y  MUL AH  MOV RESULT,AX  MAIN ENDP  END MAIN |
| RESULT=2^X | RESULT = X\*Y |

**Q1 – D - [6] Trace the following code to find AX after each command in HEX format.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **AX** | | | |
| **0** | **0** | **0** | **0** |
| MOV AX,2323H | 2 | 3 | 2 | 3 |
| ADD AX,9FC4H | C | 2 | E | 7 |
| SBB AX,098DEH | 2 | A | 0 | 9 |
| ADC AX,85FCH | B | 0 | 0 | 5 |
| MUL AL | 0 | 0 | 1 | 9 |
| XCHG AH, AL | 1 | 9 | 0 | 0 |

**Q2 – A - [15] Assuming that, BIU is twice faster than the EU and is able to buffer up to 4 commands. Put (√) in front of lines causing branch penalty. [*Note: choose carefully to avoid marks penalty*]**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| .MODEL SMALL |  |  | MOV DI, OFFSET Y |  |  | MUL BL |  |
| .STACK 64 |  | PUSH BX |  | LBL6: MOV CX,0 |  |
| .DATA |  | LBL2: POP DX |  | ADD CL, DH |  |
| X DB 6 |  | STOSB |  | JNC LBL4 | **√** |
| Y DB 9,5,7 |  | CMP AL, BH |  | LBL7: OR CL, CH |  |
| .CODE |  | LBL3: JA LBL6 | **√** | JNE LBL3 |  |
| MAIN PROC FAR |  | LBL4: XCHG DL, BH |  | JE LBL2 |  |
| MOV AX, @DATA |  | PUSHA |  | LBL8: CMP DX, BX |  |
| MOV DS, AX |  | MOV AL,3 |  | JGE LBL5 |  |
| MOV ES, AX |  | POPA |  | XOR BH, BL |  |
| MOV AL, X |  | CLC |  | JZ LBL10 | X |
| MOV BL, Y [2] |  | ADC AL, BL |  | LBL9: AND AL, DH |  |
| ADD AL, BL |  | JNC LBL5 | X | SHL AL, CL |  |
| ADC BL, Y[1] |  | MOV DX, WORD PTR Y |  | JO LBL7 |  |
| JB LBL9 |  | JNZ LBL1 |  | LBL10: HLT |  |
| LBL1: RCR BL,1 |  | LBL5: SBB DL, DH |  | MAIN ENDP |  |
| MOV BH, BYTE PTR X+2 |  | JE LBL8 | **√** | END MAIN |  |

**Q2 – B - [28] Trace the following code to find AX, BX and flags values after each command. Assume that flags values initially are SF=0,CF=0, PF=0, AC=0, OF=0 and ZF=0 and all registers are 0**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **AX** | **BX** | **SF** | **CF** | **PF** | **AF** | **OF** | **ZF** |
|  | **0000 H** | **0000 H** | **0** | **0** | **0** | **0** | **0** | **0** |
| MOV AX, 0FACEH | FACE | 0000 | 0 | 0 | 0 | 0 | 0 | 0 |
| MOV BX, 0CAFEH | FACE | CAFE | 0 | 0 | 0 | 0 | 0 | 0 |
| AND BL, AL | FACE | CACE | 1 | 0 | 0 | 0 | 0 | 0 |
| CMP AH, BH | FACE | CACE | 0 | 0 | 1 | 0 | 0 | 0 |
| SAL AL,1 | FA9C | CACE | 1 | 1 | 1 | 0 | 0 | 0 |
| MUL BX | C188 | CACE | 1 | 1 | 1 | 0 | 1 | 0 |
| OR AL, DL | C188 | CACE | 1 | 0 | 1 | 0 | 0 | 0 |

**Q3 - [12] – *Hany: Endgame***. All quality assurance teams worked with Hany decided to kill him. He told them that “*I’ve been changed; I will never write useless commands again*”. This is his last chance.

[6] a. What does this program do, knowing that, the final result is saved in AL only (nothing else matters)?

[ Find index of an element in array ]

***Note: AL final content is not required.***

[6] b. Now you have to decide. They will kill him if they detected any single line that could be removed safely or even a modified version of this code that makes exactly the same function with a lower number of lines. Noting that, it is not allowed to change any part of the data segment. **Circle around your choice**

**[ Kill Hany – Don’t Kill Hany ]**

**If you decided to kill him, you must write your shorter code; otherwise, leave the following area empty *and give him the last chance to live*.**

|  |  |
| --- | --- |
| .386  .MODEL SMALL  .DATA  X DB 5,4,7,6  Y DB 7  .CODE  MAIN PROC FAR  MOV AX, @DATA  MOV DS, AX  MOV EBX,0  S:MOV CL, [X+EBX]  CMP CL, Y  JE E  INC EBX  JMP S  E:MOV AL, BL  MAIN ENDP  END MAIN | MODEL SMALL |
| .DATA |
| X DB 5,4,7,6 |
| Y DB 7 |
| .CODE |
| MAIN PROC FAR |
| MOV AX, @DATA |
| MOV DS, AX |
| MOV BX,0 |
| S:MOV CL, [X+BX] |
| CMP CL, Y |
| JE E |
| INC BX |
| JMP S |
| E:MOV AL, BL |
| MAIN ENDP |
| END MAIN |

**Q4 - [12] An attacker intercepted a serial transmission for 25 milliseconds. The transmission was as the following table. Complete the following transmission configurations**

1. **[3] The max length between transmitter and receiver =……………ft**

**10 words -1000 ft**

1. **[4] Number of transmitted words = ………**
2. **[5] Indicate if the following configuration could be a valid transmission configuration for this intercepted transmission or not by writing (√) in front of valid possible configurations only.**

*[Note: Choose carefully to avoid marks penalty]*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Word Size | Parity | Number of stop bits | Answer |  | # | Word Size | Parity | Number of stop bits | Answer |
| 1 | 7 | No Parity | 1 | **√** |  | 11 | 7 | Low Parity Sticky | 2 |  |
| 2 | 7 | Even Parity | 1 | **√** |  | 12 | 8 | Low Parity Sticky | 2 |  |
| 3 | 8 | Odd Parity | 1 |  |  | 13 | 8 | Even Parity | 2 |  |
| 4 | 7 | Odd Parity | 1 |  |  | 14 | 7 | High Parity Sticky | 2 |  |
| 5 | 8 | High Parity Sticky | 1 |  |  | 15 | 7 | Even Parity | 2 |  |
| 6 | 7 | Low Parity Sticky | 1 |  |  | 16 | 8 | No Parity | 2 |  |
| 7 | 8 | Even Parity | 1 |  |  | 17 | 8 | Low Parity Sticky | 1 |  |
| 8 | 7 | Odd Parity | 2 |  |  | 18 | 7 | No Parity | 2 | **√** |
| 9 | 8 | High Parity Sticky | 2 |  |  | 19 | 7 | High Parity Sticky | 1 | **√** |
| 10 | 8 | Odd Parity | 2 |  |  | 20 | 8 | No Parity | 1 | **√** |

**[Note: header numbers are for the transmission order. Meaning that, the first, second and third transmitted bits are ones, the fourth is zero and the fifth is one and so on]**

First transmitted bit 8/no parity/1/11/500

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** | ***9*** | ***10*** | ***11*** | ***12*** | ***13*** | ***14*** | ***15*** | ***16*** | ***17*** | ***18*** | ***19*** | ***20*** |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
|  | | | | | | | | | | | | | | | | | | | |
| ***21*** | ***22*** | ***23*** | ***24*** | ***25*** | ***26*** | ***27*** | ***28*** | ***29*** | ***30*** | ***31*** | ***32*** | ***33*** | ***34*** | ***35*** | ***36*** | ***37*** | ***38*** | ***39*** | ***40*** |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
|  | | | | | | | | | | | | | | | | | | | |
| ***41*** | ***42*** | ***43*** | ***44*** | ***45*** | ***46*** | ***47*** | ***48*** | ***49*** | ***50*** | ***51*** | ***52*** | ***53*** | ***54*** | ***55*** | ***56*** | ***57*** | ***58*** | ***59*** | ***60*** |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
|  | | | | | | | | | | | | | | | | | | | |
| ***61*** | ***62*** | ***63*** | ***64*** | ***65*** | ***66*** | ***67*** | ***68*** | ***69*** | ***70*** | ***71*** | ***72*** | ***73*** | ***74*** | ***75*** | ***76*** | ***77*** | ***78*** | ***79*** | ***80*** |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
|  | | | | | | | | | | | | | | | | | | | |
| ***81*** | ***82*** | ***83*** | ***84*** | ***85*** | ***86*** | ***87*** | ***88*** | ***89*** | ***90*** | ***91*** | ***92*** | ***93*** | ***94*** | ***95*** | ***96*** | ***97*** | ***98*** | ***99*** | ***100*** |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|  | | | | | | | | | | | | | | | | | | | |
| ***101*** | ***102*** | ***103*** | ***104*** | ***105*** | ***106*** | ***107*** | ***108*** | ***109*** | ***110*** | ***111*** | ***112*** | ***113*** | ***114*** | ***115*** | ***116*** | ***117*** | ***118*** | ***119*** | ***120*** |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |