

Core competency test for ECE 372/373

N. Natarajan

1 Basics

You should be able to answer the following given sufficient time. In the test, you will be asked to answer 10 such questions in 20 minutes (average of 2 minutes a question. Some may take longer but most should not take more than a minute or so.) The test is a closed book test and you will be allowed to bring only the Motorola reference books.

1. Convert the following decimal numbers into 16 bit *signed* hexadecimal numbers:
100, 272, -1, -2, -255
2. Convert the following 8-bit signed hexadecimal numbers to decimal: **\$AC**, **\$60**, **\$FF**, **\$22**, **\$04**
3. *10 points* Perform the following 8-bit addition and indicate which of the **C**, **V**, **N**, **Z** flags will be set

$\$20 + \70	<i>Answer</i> =	$C =$	$N =$	$Z =$	$V =$
$\$2A + \$8F$	<i>Answer</i> =	$C =$	$N =$	$Z =$	$V =$
$\$5A + \$A6$	<i>Answer</i> =	$C =$	$N =$	$Z =$	$V =$
$\$A0 + \$8C$	<i>Answer</i> =	$C =$	$N =$	$Z =$	$V =$

4. Increment by 5 a 16 bit number stored in locations \$00-\$01.
5. Decrement by 7 a 16 bit number stored in locations \$00-\$01.
6. Add the 16 bit number stored in locations \$00-\$01 to the 16 bit number stored in locations \$02-\$03 store the result back in locations \$00-\$01
7. Shift left (once) a 16 bit number stored in locations \$00-\$01.
8. Compare a 16 bit unsigned number stored in locations \$00-\$01 with an unsigned number stored in locations \$02-\$03 and set B register to \$00 if they are the same, \$01 if the first number is larger, \$FF if the first number is smaller

9. Set register B to \$FF if both bits #2 and #5 in register A are set. You must use the *MASK* equated as

```
MASK EQU \%00100100
```

10. Set register B to \$FF if either bits #2 and #5 in register A are set. You must use the *MASK* equated as

```
MASK EQU \%00100100
```

11. Set register B to \$FF if exactly one of the bits #2 and #5 in register A are set. You must use *MASK*, *BIT5*, *BIT2* equated as

```
MASK EQU \%00100100
BIT5 EQU \%00100000
BIT2 EQU \%00000100
```

12. Use a counting loop to subtract the contents of register Y from register X; at the end of the loop, X should contain X-Y.

13. Test if any of the numbers in locations \$D000-\$D03F is zero. If they are all non-zero, set register A to any non-zero number. If atleast one of them is zero, set register A to 0. You use the following equates:

```
START EQU \%D000
LENGTH EQU \%40
```

14. Write a *function* that will read a character from the input without echoing it. The function should loop until it receives one of the following valid characters: a A b B c C d D. The function should return the character in the accumulator A.

15. Write a *function* that will set bit #4 in memory location \$10 without affecting other bits in that location.

16. Write a *function* that will combine the values in locations \$10 and \$11 and store the value in location \$12. The first four bits of the results (bits 7-4) should come from last four bits in location \$10 and the last four bits of the result (bits 3-0) should come from the last four bits in location \$11

17. Write a *function* that will turn on the A/D convertor and wait for 400 microseconds before returning.

18. Write a *function* that will read the value in the A/D channel #2 and return the value in accumulator **B**.
19. Write a *function* that will configure port D so that pins #4, and #3 are output pins and all others are input pins
20. Write a *function* that output the last 4 bits (bits 3-0) of the value in accumulator B to the pins 5-2 in port D.
21. Write a *function* that will set the scaler for RTI interrupt so that the interrupt occurs at the fastest rate. This function should also enable the RTI interrupt before returning.
22. Write a *interrupt service routine* that will service the RTI interrupt and, as part of the service, decrement a 16 bit value stored in locations \$10-\$11.
23. Write a *interrupt service routine* that will service the OC2 interrupt and as part of the service, schedule the next OC2 interrupt to occur \$2500 clock ticks after the current interrupt (the interrupt that is being serviced).
24. Write a *function* that will read a character from the input without echoing it. The function should loop until it receives one of the following valid characters: a A b B c C d D. The function should return the character in the accumulator A.
25. Write a *function* that will set bit #4 in memory location \$10 without affecting other bits in that location.
26. Write a *function* that will combine the values in locations \$10 and \$11 and store the value in location \$12. The first four bits of the results (bits 7-4) should come from last four bits in location \$10 and the last four bits of the result (bits 3-0) should come from the last four bits in location \$11
27. Write a *function* that will turn on the A/D convertor and wait for 400 microseconds before returning.
28. Write a *function* that will read the value in the A/D channel #2 and return the value in accumulator **B**.
29. Write a *function* that will configure port D so that pins #4, and #3 are output pins and all others are input pins
30. Write a *function* that output the last 4 bits (bits 3-0) of the value in accumulator B to the pins 5-2 in port D.
31. Write a *function* that will set the scaler for RTI interrupt so that the interrupt occurs at the fastest rate. This function should also enable the RTI interrupt before returning.

32. Write a *interrupt service routine* that will service the RTI interrupt and, as part of the service, decrement a 16 bit value stored in locations \$10-\$11.
33. Write a *interrupt service routine* that will service the OC2 interrupt and as part of the service, schedule the next OC2 interrupt to occur \$2500 clock ticks after the current interrupt (the interrupt that is being serviced).
34. Write a *function* that will increment a 16 bit number stored in locations \$0010-\$0011. The incremented value should be stored back in the same locations.
35. Write a *function* that will add the value in register B to a 16 bit number stored in locations \$0010-\$0011 and store the answer back in the same locations.
36. Write a *function* that will test the value in location \$1003 and set the **B** register to \$07 if the bits #0 and #1 are both set. Or else it should load the value \$11 in the **B** register.
37. Write a *function* that will test the value in location \$1003 and set the **B** register to \$07 if at least one of the bits #0 and #1 are set. Or else it should load the value \$11 in the **B** register.
38. Write a *function* that will test the value in location \$1003 and set the **B** register to \$07 if none of the bits #0 and #1 are set. Or else it should load the value \$11 in the **B** register.
39. Write a *function* that will test the value in location \$1003 and set the **B** register to \$07 if exactly one of the bits #0 and #1 are set. Or else it should load the value \$11 in the **B** register.
40. Write a *function* that will add a 64 bit number stored in locations \$00-\$07 to a 64 bit number stored in locations \$10-\$17 and store it as a 64 bit number stored in locations \$20-\$27.
41. Write a *function* that will count the number of times the value \$0A occurs in the memory locations \$00-\$0F. The function should return the answer in the **B** register.