

**TUTORIAL 1 - QUESTIONS ON BASIC COMPUTING**

What do we call the position in memory where a value is stored? \_\_\_\_\_

What does the anonym "CPU" stand for? \_\_\_\_\_  
\_\_\_\_\_

What does the anonym "RAM" stand for? \_\_\_\_\_  
\_\_\_\_\_

Which of the following is true of the Program Counter (PC)?  
(a) It stores the intermediate value in arithmetic operations ( )  
(b) The value of the PC points to the next instruction to be executed ( )  
(c) The value of the PC is a copy of the next instruction to be executed ( )  
(d) The PC value is usually incremented after execution of an instruction ( )  
(e) The PC contains the address of a memory location held in a register ( )

Which of the following is true?  
(a) The instruction register contains the address of the next instruction ( )  
(b) The instruction register contains the value of the current instruction ( )  
(c) The instruction register contains the address of the current instruction ( )  
(d) The instruction register contains the size of the current instruction ( )

Which of the following statements are true about the MOV instruction?  
(a) The MOV instruction is stored as a number in the computer ( )  
(b) It takes two operands which indicate the source and destination ( )  
(c) The instruction clears the value at the source address after execution ( )  
(d) The instruction increments the destination after execution ( )  
(e) The instruction modifies the value stored at the destination ( )

Which of the following is true of the instruction MOV r0, #3?  
(a) The destination value is to be placed in R0 ( )  
(b) The source value is contained in register 3 ( )  
(c) The source value is the value 3 ( )  
(d) The original value in R0 is overwritten and no longer saved ( )

The following is fragment of an assembly language program:  
MOV r0, #3  
ADD r0, #4           What is the final value in register r0? \_\_\_\_\_  
SHIFTL r0,4  
OR r0,#1           Which one instruction needs to be deleted  
ADD r0,1           for the final value in r0 to be 0x08? \_\_\_\_\_

Examine the following machine code instructions:  
MOV r1,#5           What is the value in register r0 after execution? \_\_\_\_\_  
MOV r0,#2  
ADD r1,r0           What is the value in register r1 after execution? \_\_\_\_\_

Is the Harvard Architecture more common than the Von Neuman Architecture? \_\_\_\_\_

The Motorola Power PC is the processor at the heart of modern Macintosh computers:  
Does the Motorola Power PC use the Harvard Architecture? \_\_\_\_\_

The Intel Pentium PC is the processor at the heart of modern PC computers:  
Does a Pentium PC use the Harvard Architecture? \_\_\_\_\_

The PIC is the processor at the heart of many domestic appliances:  
Does the PIC Microcontroller use the Harvard Architecture? \_\_\_\_\_

## TUTORIAL 2 - OPERATION OF THE CPU

Write the hexadecimal values of the following expressions:

$\sim 0x0010\ 1010 + 1$  \_\_\_\_\_

$0x05 \gg 2$  \_\_\_\_\_

$0x0000\ 0002 \ll 9$  \_\_\_\_\_

$0x0567\ 0567 \& 0x5555\ 3333$  \_\_\_\_\_

$0x8000\ 8001 \text{ OR } 0x0001\ 0001$  \_\_\_\_\_

$0xFFFF\ FFFF + 1$  \_\_\_\_\_

$0xFFFF\ FFF8 \text{ XOR } -1$  \_\_\_\_\_

The variable fred is a 4 byte signed integer with value  $0xFFFF\ FFF7$

Is the number positive? \_\_\_\_\_

What is the value? \_\_\_\_\_

What is the value of fred +2? \_\_\_\_\_

The following fragment of a C program takes an integer b and calculates a value which it stores in an integer c. What mathematical function does this perform?

$c = (b \ll 2) + b;$  \_\_\_\_\_

When the processor executes `MOV r0,0x0080` which of the following values appear on the data bus during fetch and execution of the instruction?

- (a) The instruction code for `MOV r0,0x0080` ( )
- (b) The value stored at memory location `0x0080` ( )
- (c) The address of a location containing `MOV r0,0x0080` ( )
- (d) The value stored in register `r0` ( )
- (e) The value of the Stack Pointer (SP) register ( )

If `r0` contains three and the following instructions are executed what is the value in `r0` after execution?

```
skip:
    SUB r0, #1
    BRA.NE skip
    ADD r0, #1
```

\_\_\_\_\_

(BRA.NE is the mnemonic for BRANCH on NOT EQUAL to zero)

Examine the following small C program.

```
main()
{
    int c,z;
    c=5; z=1;
loop: c++; z++; /* decrement count */
    printf("c == %d 0x%x\n",count,count);
    if (c < 8) {goto loop;}
}
```

What is the first number printed? \_\_\_\_\_

What is the last number printed? \_\_\_\_\_

what is the final value of z? \_\_\_\_\_

### TUTORIAL 3 MEMORY AND ADDRESSES

Which of the following may be written to by setting the R/W wire to false?  
RAM, ROM, FLASH, EPROM

\_\_\_\_\_

Which of the following loose all data when power is removed?  
RAM, ROM, FLASH, EPROM

\_\_\_\_\_

Which of the following may be used to store programs?  
RAM, ROM, FLASH, EPROM

\_\_\_\_\_

What does the E in EPROM stand for?

\_\_\_\_\_

Which of the following statements is true about caches?

( )

(a) caches may be accessed much faster than main memory

( )

(b) caches do not loose data when the power is removed

( )

(c) read caches contain duplicate copies of data in main memory

( )

(d) write caches contain duplicate copies of data in main memory

( )

(e) cache memory is more expensive than standard RAM

( )

What does the value \*a evaluate to:

(a) The value of a

( )

(b) The number of bytes stored at a

( )

(c) The address pointed to by a

( )

(d) The address of the byte pointed to by a

( )

(e) The address of the address pointed to by a

( )

What does the C statement "value \*(&a)" evaluate to:

(a) The value of a

( )

(b) The number of bytes stored at a

( )

(c) The address pointed to by a

( )

(d) The address of the byte pointed to by a

( )

(e) The address of the address pointed to by a

( )

Examine the following code fragment:

```
{
    char a,b,c;
    int d;
    ....
    c = a;
    if (c < b) {c=b;}
    if (c < d) {d=c;}
    ....
}
```

How many bytes are allocated to variable a?

\_\_\_\_\_

How many bytes are allocated to variable d?

\_\_\_\_\_

Given the initial values of: a=1,b=2,c=3,d=4

What is the final value of c (in hexadecimal)?

\_\_\_\_\_

What is the final value of d (in hexadecimal)?

\_\_\_\_\_

Given the initial values of: a=4,b=3,c=2,d=1

What is the final value of c (in hexadecimal)?

\_\_\_\_\_

What is the final value of d (in hexadecimal)?

\_\_\_\_\_

Given the initial values of: a=-1,b=-3,c=0,d=0

What is the final value of c (in hexadecimal)?

\_\_\_\_\_

What is the final value of d (in hexadecimal)?

\_\_\_\_\_

## TUTORIAL 4 - STACK OPERATIONS

When the processor executes PUSH r0 which of the following values appear on the address bus during fetch and execution of the instruction?

- (a) The instruction code for PUSH r0 ()
- (b) The address of a location containing the value in the Stack Pointer (SP) ()
- (c) The address of a location containing PUSH r0 ()
- (d) The value stored in register r0 ()
- (e) The value of the Stack Pointer (SP) register ()

Which of the following statements is true about stacks?

- (a) Stacks are stored in main memory ()
- (b) The Stack Pointer points to the location of the first value written to the stack ()
- (c) The Stack Pointer is incremented by a push byte operation ()
- (d) The Stack Pointer is incremented by a pop byte operation ()
- (e) The Stack Pointer contains the value of the last item written to the stack ()

Observe the following program fragment, and calculate the contents of r0 and r1

```

MOV r1,#1
MOV r0,#3
PUSH r0          What is the final value in r0?      _____
MOV r0,#5
PUSH r0
ADD r0,#5        What is the final value in r1?      _____
POP r0
ADD r0,r1
POP r1
    
```

The following procedure is called with a=1 and b=2:

```

int diff(a,b)
int a,b;
{
    return(a-b);
}
    
```

Given the previous Stack Pointer was 0x80 01, what is the new value of the SP, assuming an int is 4B and the SP and Stack Frame pointer (SF) are each also 4B? \_\_\_\_\_

What is the new value of SF? \_\_\_\_\_

Consider the following C program:

```

void z(a,b)
int a,b;
{
    int c;
    c =a; a =b; b =c;
    return;
}
main()
{
    int y0,y;
    y0 = 10; y = 0x010;
    printf("a) %d\n",y);
    z(y0,y);
    printf("b) %d\n",y);
}
    
```

What is the value printed after a)? \_\_\_\_\_

What is the value printed after b)? \_\_\_\_\_

In procedure Z what was the final value of c? \_\_\_\_\_

## TUTORIAL 5 - ARRAYS

Which statements are true about the array "char miserable[10]"?

- (a) The array starts at location miserable
- (b) The array ends at location miserable+10
- (c) The value miserable[10] accesses an illegal memory location
- (d) The array may be used to store a string
- (e) The value of \*miserable is the same as miserable[0]

Consider the following C example:

```
void t(i,k)
  int *i,*k;
{
  int c;
  c =*i; *i=*k; *k=c;
  return;
}
main()
{
  int c, s;
  int item[4];
  item[0]=5;item[1]=2; item[2]=2;item[3]=3;
loop:   c = 3; s = 0;
next:  if(item[c] < item[c-1]){ t(item+c,item+(c-1)); s=1; }
      c--; if(c > 0){ goto next;}
      printf("item[%d] == %d\n",count,item[c]);
      if(s != 0) { goto loop;}
}
```

- Does the procedure "t" have a return value? \_\_\_\_\_
- How many parameters does the procedure "t" accept when it is called? \_\_\_\_\_
- What is the final value in item[3]? \_\_\_\_\_
- What is the last value printed by this program? \_\_\_\_\_

When a C program executes "printf("Hello World\n")" which of the following are true?

- (a) The \n indicates that the computer should output a new line character?
- (b) The C program prints the message when the program is linked
- (c) The program calls a print routine which passes the data to the operating system
- (d) The characters are printed immediately by the operating system
- (e) The operating system sends the characters one at a time to an output device

What does the message "segmentation fault" mean when displayed by a computer?

- (a) There is a hardware error with a segment of the memory
- (b) A program attempted to write to the memory reserved from the program itself
- (c) A program has attempted to modify the stack pointer
- (d) The program has a missing ; or }
- (e) The program has attempted to read a variable from the static memory area

Which of the following are true about compilers and assemblers?

- (a) The linker is a piece of software
- (b) A compiler is simpler than an assembler because it works at a higher level
- (c) Assembly language is more often used than C
- (d) The C language is source code portable between different types of computer
- (e) An executable C program may execute on any type of CPU

Which of the following are true about computers?

- (a) The registers in a computer are formed from flip-flops and logic gates
- (b) The computer bus is formed from flip-flops and logic gates
- (c) A ROM uses flip-flops to store data
- (d) A Static-RAM uses flip-flops to store data
- (e) The data bus must use tri-state logic because it both sends and receives data