

程序代写代做 CS 编程辅导

COMP2300/6300

Computer Organisation and Programming



Revision

Dr Charles Martin

Semester 1, 2022



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Admin Time

Assignment 2 done!

Lab 12 this week! Make your own OS!

Lab Pack 4 due: Wednesday 1/6/2022 23:59 AEST (day before exam period)

Quiz 2 still open!

Prepare for your final exam!

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



It's been a journey...

程序代写代做 CS 编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



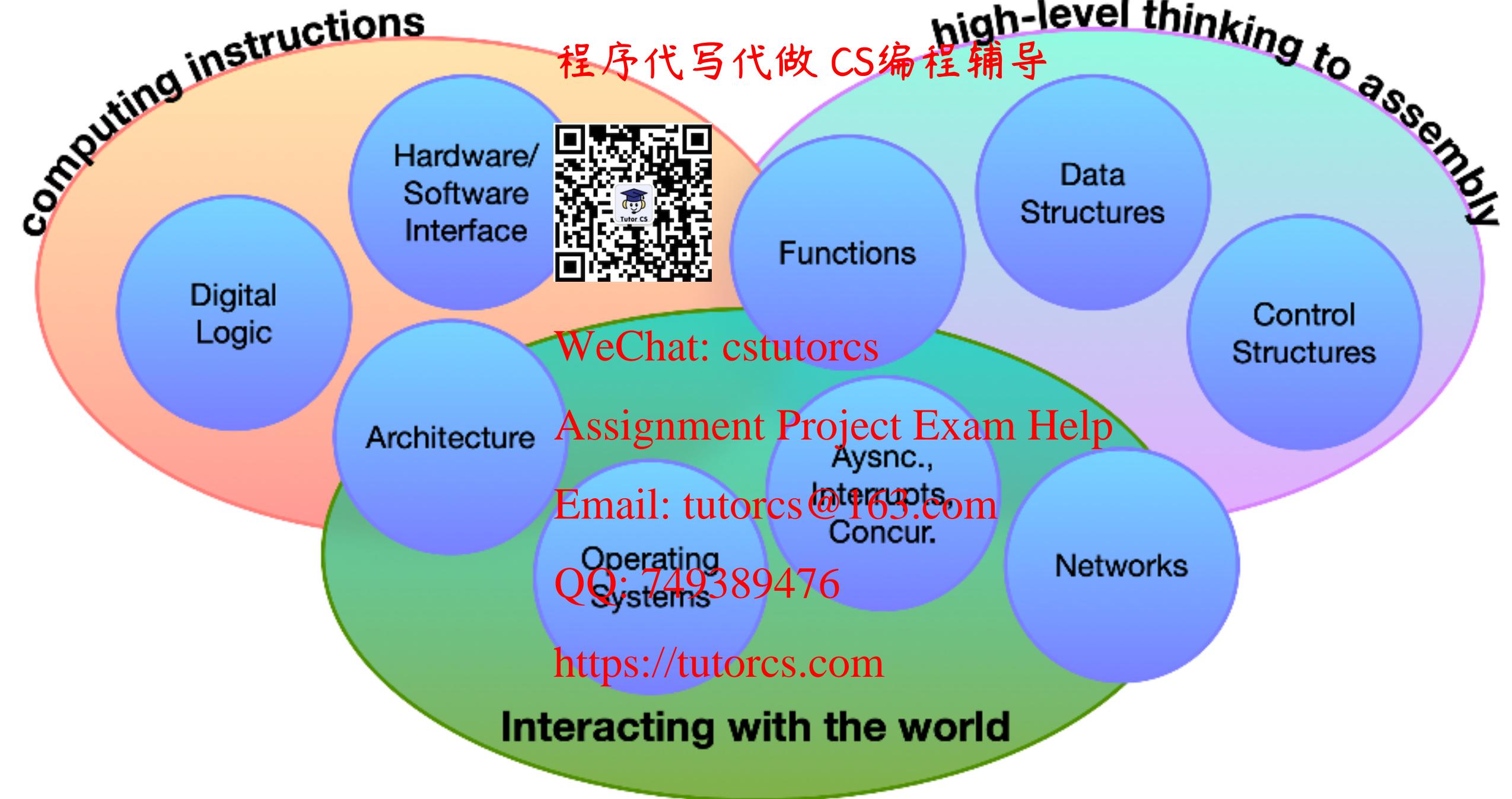
WeChat: cstutorcs

What was this course about again?
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Digital Logic

Digital Logic Topics (in short):

- Boolean Algebra
- Combinatorial Logic Functions:



- Digital Electronics - Logic Gates

WeChat: cstutorcs

- Binary Encoding, and 2's-Complement

- Adders: Half, Full, Ripple Carry

Assignment Project Exam Help

- Arithmetic Logic Unit

Email: tutorcs@163.com

- Simple CPU Architecture

QQ: 749389476

<https://tutorcs.com>

Logic gates

程序代写代做 CS编程辅导



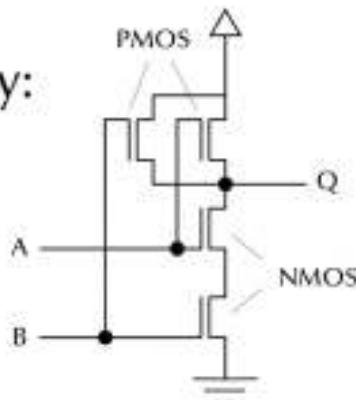
Symbolic: $Q = \overline{A \wedge B}$

Diagram:



≡

Technology:



Elementary logic gate symbols:
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



1. how many bits can be added together?
2. how long does it take?



3. where does the final carry bit go?

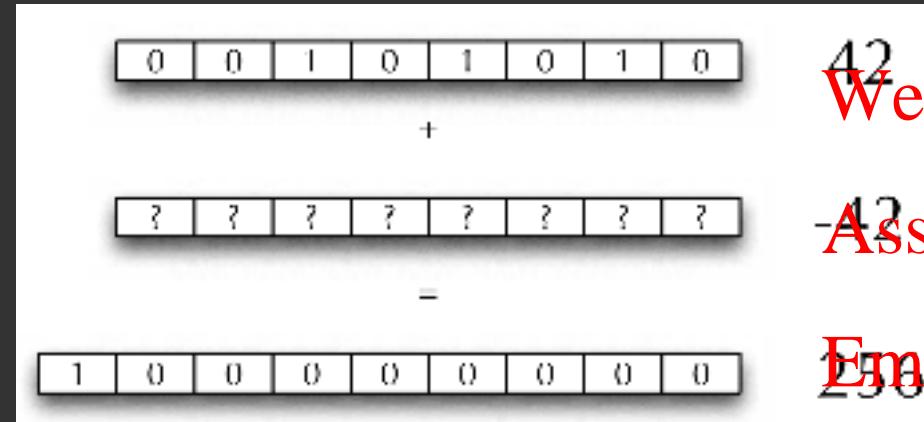
WeChat: cstutorcs



Twos complement representation

程序代写代做 CS 编程辅导

The basic idea: define (binary) negative numbers so the adder works.



42
WeChat: cstutorcs

-42
Assignment Project Exam Help

256
Email: tutorcs@163.com

QQ: 749389476

How do we make a number negative? Invert bits and add one! Why does that work?

<https://tutorcs.com>

Flip-flops

程序代写代做 CS编程辅导



WeChat: cstutorcs
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

What's the difference between Boolean algebra, combinational logic, and sequential logic?



Why do we need all three of these to describe computers?

Are there parts of the story missing from COMP2300?

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

CPU Architecture

程序代写代做 CS编程辅导

What are the **main components**?



Can you **explain** what each of the components do?

We come back to this later...

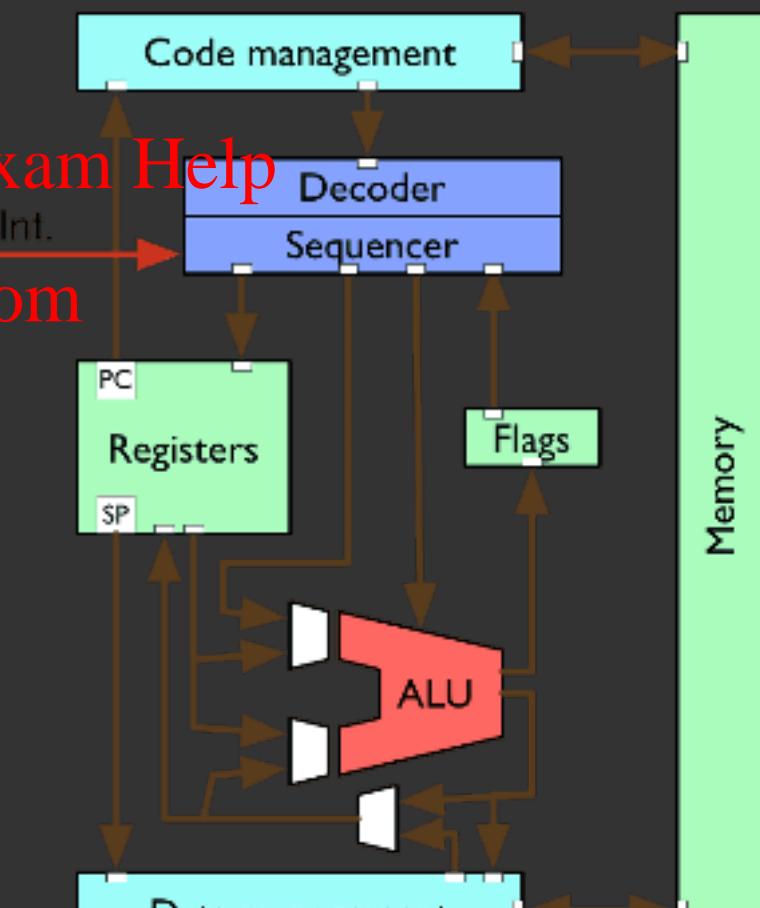
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



程序代写代做 CS编程辅导



WeChat: cstutorcs

Hardware/Software Interface

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Hardware/Software Interface Topics (in short)

- Structure of an instruction
- Assembly to CPU instructions
- CPU Status Flags (NCZV)
- ARM v7 instructions (adding, subtracting, moving, rotate/shift, bit-wise ops)
- loading and storing from memory
- branch instructions
- Contents of Quick Ref. Card



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

NCZV Flags

- Negative
- Zero
- Carry
- Overflow

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Negative

程序代写代做 CS编程辅导

This status flag is set when the result of an ALU operation is negative *if interpreted as a two's complement signed integer*



```
movs r0, 5  
movs r1, 6  
subs r2, r0, r1
```

don't forget the **s** suffix

ALU operation is negative *if interpreted as a two's complement signed integer*

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Zero

程序代写代做 CS编程辅导

This status flag is set when the result of an ALU operation is zero



```
movs r5, 5  
movs r6, -5  
adds r4, r5, r6
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Carry

程序代写代做 CS编程辅导

This status flag is set when the result of an ALU operation requires a “carry out” if interpreted as an unsigned 32-bit integer (i.e. it requires 33 or more bits to represent)



```
movs r2, 0xFF000000  
movs r3, 0xFF000000  
adds r5, r2, r3
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Overflow

程序代写代做 CS编程辅导

This status flag is set when the result of an ALU operation would overflow the min/max value if interpreted as a twos complement integer



```
movs r0, 0x7FFFFFFF @ largest signed integer  
adds r0, 1
```

```
movs r0, 0x80000000 @ smallest signed integer  
subs r0, 1
```

Assignment Project Exam Help

smallest signed integer

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

```
movs r0, 5  
movs r1, 6  
subs r2, r0, r1
```



What flags will be set after the `subs` instruction is executed?

WeChat: cstutorcs

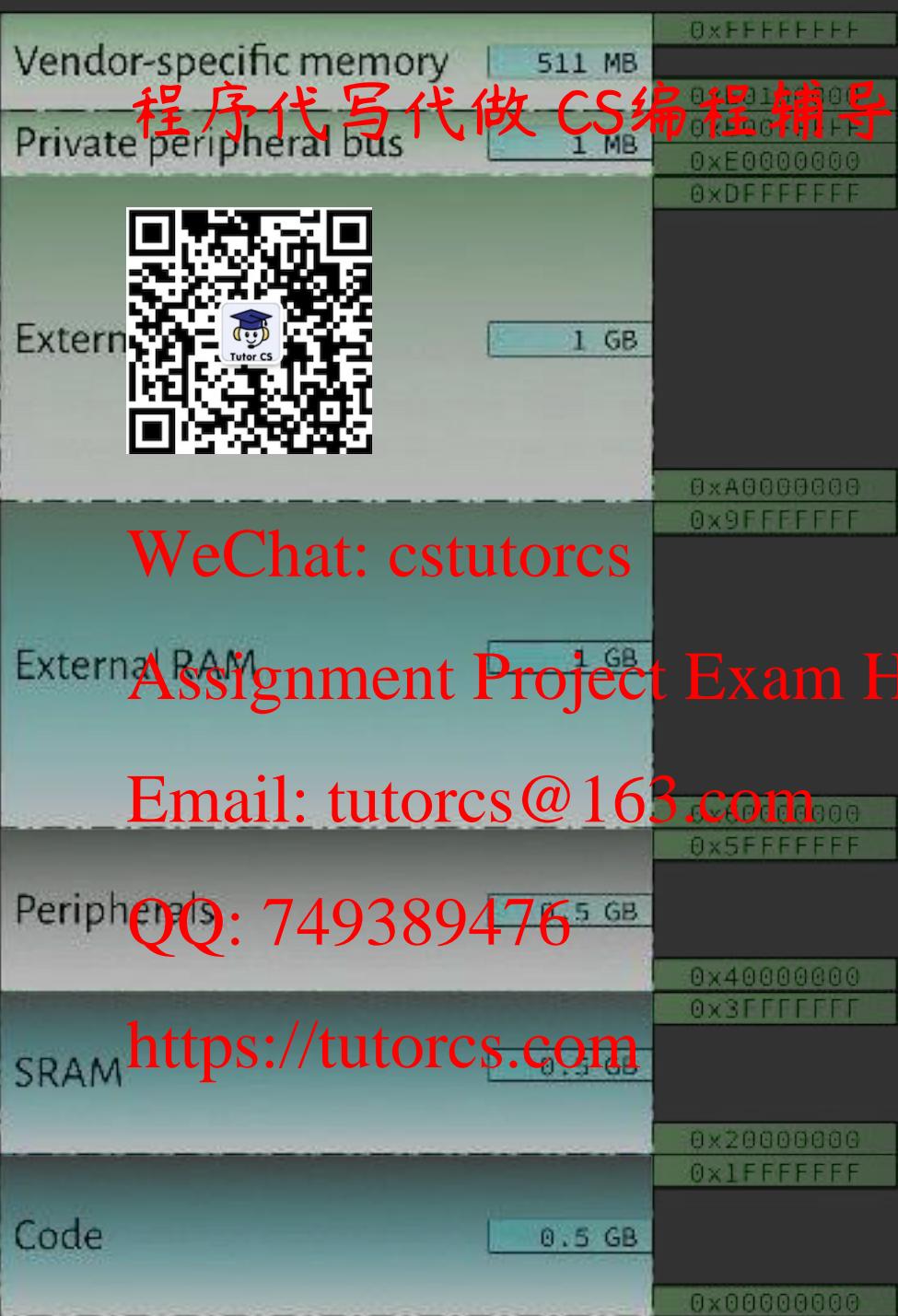
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

address range



lsl

程序代写代做 CS 编程辅导

lsr

asr

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

ror

<https://tutorcs.com>

rrx

ARM is a load-store architecture

Instructions are *either*:

- ALU operations which take inputs from registers and write results to registers, or,
- memory access operations which *just* save and load from memory



WeChat: cstutorcs

What does this mean for the programmer?

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Loading and Storing

程序代写代做 CS编程辅导

Load:



```
mov r1, 0x20000000 @ put the address in r1  
ldr r0, [r1]          @ load the data into r0
```

Store:

```
mov r0, 42  
mov r1, 0x20000000  
str r0, [r1]
```

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Extra Operations

程序代写代做 CS编程辅导

Load less than 32 bits



ldr@ load byte from register
ldrh@ load halfword from register
strb@ store byte to register
strh@ store halfword to register

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

stmdb <Rs>! , {Rgstrs} @ store multiple decrement before
ldmia <Rs>! , {Rgstrs} @ load multiple increment after
push {Rgstrs}
pop {Rgstrs}

Negative Stack

Conditional branch examples

```
beq <label> @ branch i  
bne <label> @ branch i  
bcs <label> @ branch i  
bcc <label> @ branch if C = 1  
bmi <label> @ branch if N = 1  
bpl <label> @ branch if N = 0  
bvs <label> @ branch if V = 1  
bvc <label> @ branch if V = 0
```



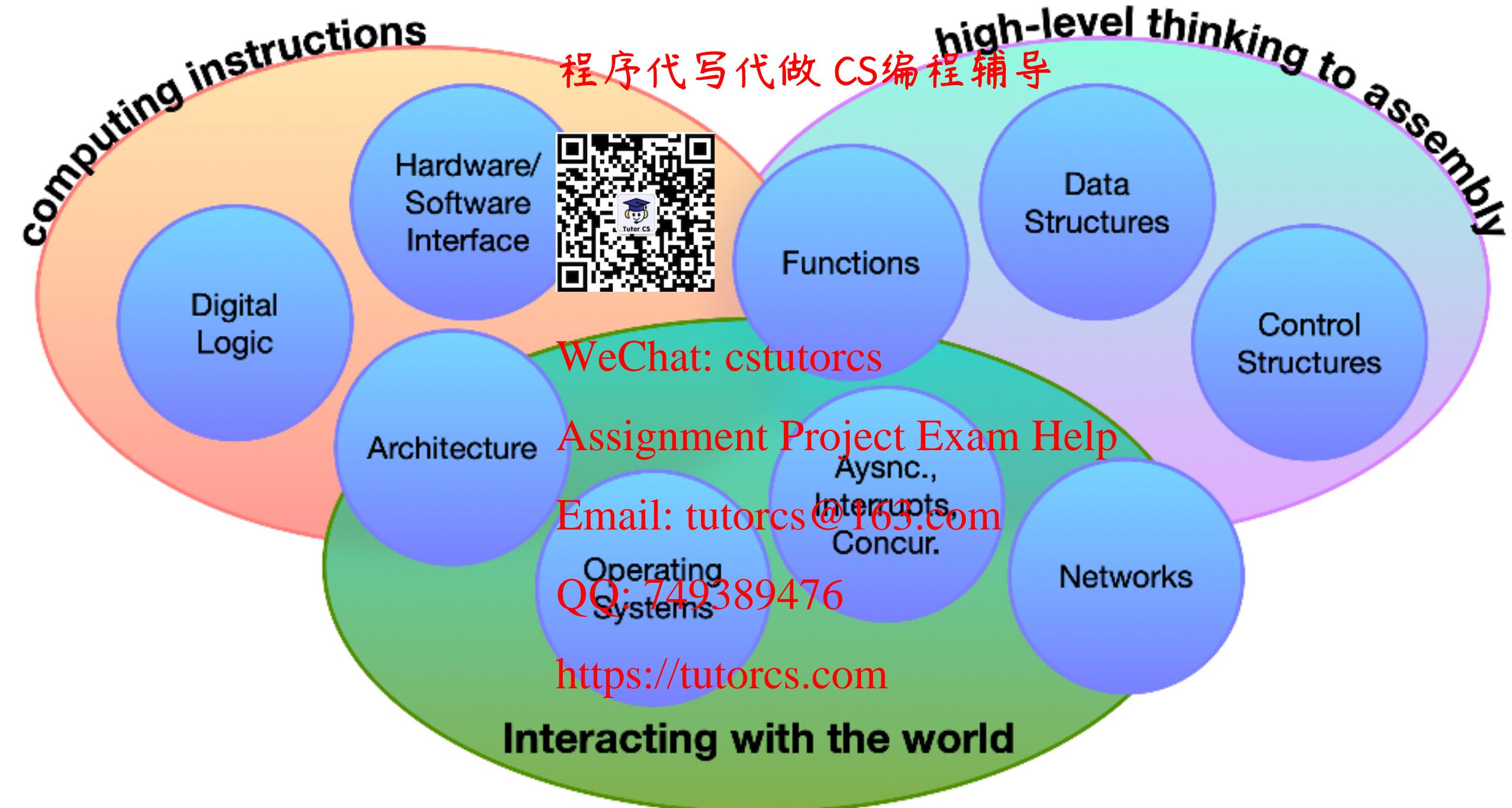
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Functions

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Functions Main Topics (in short)

- There and back again, `b1`, `b2`



- The stack

- Calling conventions

WeChat: cstutorcs

- Functions calling functions

Assignment Project Exam Help

- Functions calling themselves! (a.k.a recursive functions)

Email: tutorcs@163.com

- Local variables, and the stack frame (incl. `sp` and `fp`)

QQ: 749389476

- Relative addressing

<https://tutorcs.com>

there, and back again

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

The **ARMv7 Architecture Procedure Call Standard** is the convention we'll (try to) adhere to in programming our microbits



The full standard is quite detailed, but the general summary is:

WeChat: cstutorcs

- **r0 - r3** are the parameter and scratch registers
- **r0 - r1** are also the result registers
- **r4 - r11** are callee-save registers
- **r12 - r15** are special registers (Q, ip, sp, lr, pc)

<https://tutorcs.com>

Store and Load to the stack

```
@ Push the value in r2 onto  
str r2, [sp, -4]  
sub sp, sp, 4
```



```
@ Different one-liners for Push  
str r2, [sp, -4]!  
push {r2}  
stmdb sp!, {r2}
```

```
@ Pop the value from the "top" of the stack into r3  
ldr r3, [sp]  
add sp, sp, 4
```

```
@ One-liners for Pop  
ldr r3, [sp], 4  
pop {r3}  
ldmia sp!, {r2}
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

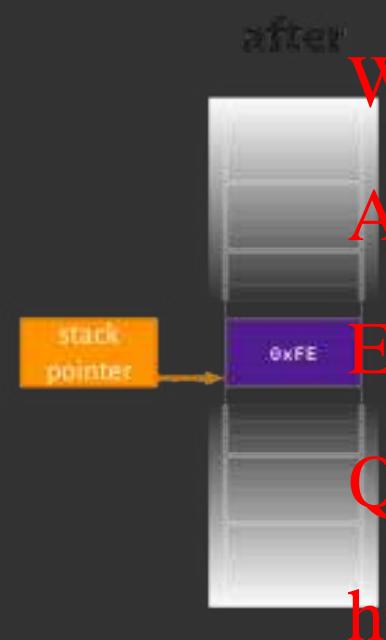
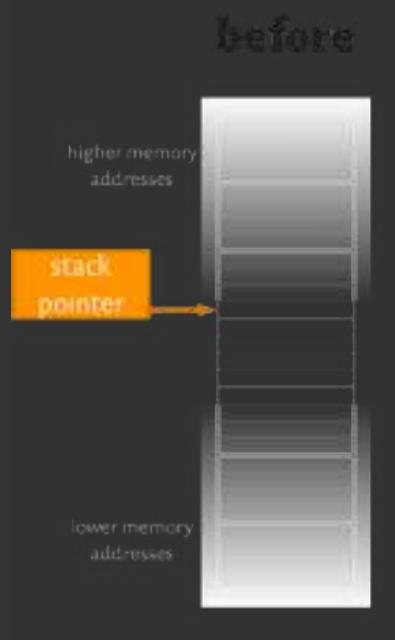
<https://tutorcs.com>

Push and Pop; illustrated

程序代写代做 CS 编程辅导

<- Push

Pop ->



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

r2

r3

sp

0xFE

r2

r3

sp

0xFE

r2

r3

sp

0xFE

r2

r3

sp

0xFE

sp

0xFE

Passing by Copy or Reference

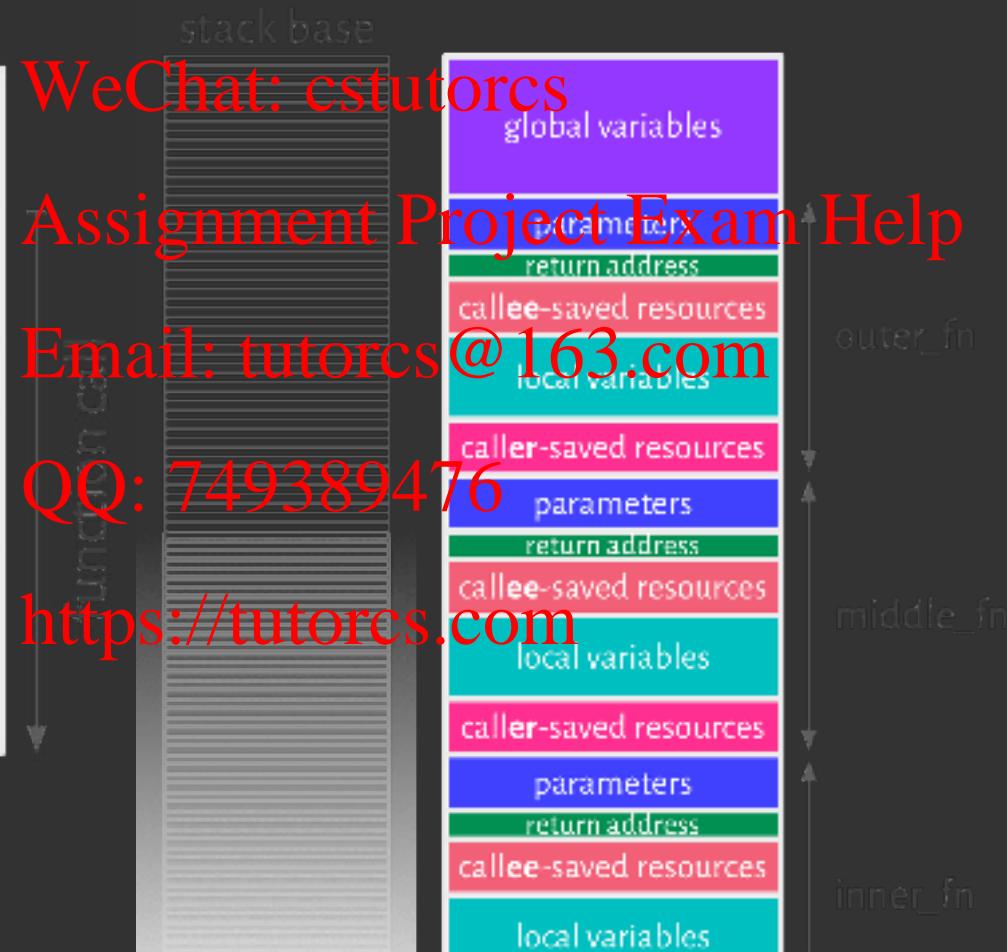


Information flow	Access		
	in	by copy by value	by reference
	out	by result	by reference (immutable)
	Parameter becomes a constant inside the function or is copied into a local variable.	No write access is allowed while the function runs (also from outside the function).	Calling function expects the parameter value to appear in a specific space at return.
in & out	Parameter is copied to a local variable and copied back at return.	by reference (mutable)	Function can read and write at any time. Outside code shall not write.

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
<https://tutorcs.com>

Function stack frame

程序代写代做 CS编程辅导



Recursive Functions: Factorial

```
fact: @ assume input is in r0  
    push {lr}  
    cmp r1, #1  
    beq base_case  
  
    @ recursive case  
    push {r1}  
    sub r1, #1  
    bl fact @ get fact(n-1)  
    pop {r1}  
    mul r0, r0, r1 @ calc fact(n-1) * n  
    b continue_code  
  
base_case:  
    mov r0, #1  
continue_code:  
    pop {lr}  
    bx lr
```



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Control Structures

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Control Structures Main Topics (in short) 程序代写代做 CS 编程辅导

- Conditional branching
- Control Structures in Machine
- if
- while
- for



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

CPSR table

<c>	meaning
eq	equal
ne	not equal
cs	carry set
cc	carry clear
mi	minus/negative
pl	plus/positive
vs	overflow set
vc	overflow clear
hi	unsigned higher
ls	unsigned lower or same
ge	signed greater or equal
lt	signed less
gt	signed greater
le	signed less or equal

程序代写代做 CS编程辅导



flags
Z=1
Z=0
C=1
C=0
N=1
N=0
V=1
V=0
C=1 \wedge Z=0
C=0 \vee Z=1
N=V
N \neq V
Z=0 \wedge N=V
Z=1 \vee N \neq V

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

while loop components

程序代写代做 CS编程辅导

```
while Register_1 < 100 loop  
    Register_1 := Register_1 ** 2;  
end loop;
```

```
while (register1 < 100) {  
    register1 = register1 * register1;  
}
```

```
while register1 < 100:  
    register1 = register1 ** 2
```

```
while Register_1 < 100 do  
    Register_1 := Register_1 ** 2;
```

```
while Register_1 < 100 do  
    Register_1 = Register_1 ** 2  
enddo
```

```
while (Register_1 < 100) {  
    Register_1 = Register_1 ** 2;  
}
```



- 1. an expression (if)
- 2. the loop condition (if)
- 3. code inside the loop

WeChat: cstutorcs
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

control structures gallery - practice these! 程序代写代做 CS编程辅导



WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
<https://tutorcs.com>

```
if "cmp r1, r2", eq, "mov r3, #1", "mov r3, #0"
for r1, 1, 100 "add r3, r1"
    mov r1, #1
    for:
        cmp r1, #100
        bgt end_for
        add r3, r1
        add r1, #1
        b for
    end_for:
        cmp r1, #100", lt, "mul r1, r1"
b while_condition
while:
    mul r1, r1
    while_condition:
        cmp r1, #100
        blt while
```

Which control structures were used in your assignments?



Is there anything you can do in a situation that goes beyond “typical control structures”?

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Data Structures

Data Structures Main Topics (in short)

- Arrays
- Structure
- Alignment
- Addressing
- Iterators
- Copying



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

array index 7		程序代写代做 CS 编程辅导	0x2000001c
array index 6			0x20000018
array index 5			0x20000014
array index 4		WeChat: cstutorcs	0x20000010
array index 3		Assignment Project Exam Help	0x2000000c
array index 2		Email: tutorcs@163.com	0x20000008
array index 1		QQ: 749389476	0x20000004
array index 0		https://tutorcs.com	0x20000000

How do we know how big an array occupies in memory?



Is it possible to write outside the boundaries of the array?

Can you make an array where the size can be changed? (mutable array?)

WeChat: cstutorcs

How do we address a particular element in an array?

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Array addressing

程序代写代做 CS 编程辅导

	7	4	3	0	
x + 0·es				e0_b0	
x + 1·es				e0_b1	
x + 2·es				e0_b2	
x + 3·es				e0_b3	
x + 4·es				e1_b0	
x + 5·es				e1_b1	
x + 6·es				e1_b2	
x + 7·es				e1_b3	
x + 8·es				e2_b0	
x + 9·es				e2_b1	
x + 10·es				e2_b2	
x + 11·es				e2_b3	
x + 12·es				e3_b0	
x + 13·es				e3_b1	
x + 14·es				e3_b2	
x + 15·es - 1				e3_b3	

Lower array bound

Can arrays always be stored “packed”?

Element size (es)
4 bytes

Upper array bound

Maybe be good to have $es = 2^n$?

What happens if array bounds are violated?

☞ And who is checking that?

$x + i \cdot es$

Add up the numbers in an array



```
ldr r0, =array @ base address  
mov r1, 0 @ from_index  
mov r2, 7 @ to_index  
@ setup  
mov r3, 0 @ "accumulator" register  
mov r4, 4 @ element size
```

WeChat: cstutorcs

```
array_sum:  
    mul r5, r1, r4 @ calculate offset  
    ldr r6, [r0, r5] @ load from offset  
    add r3, r6 @ update accumulator  
    add r1, 1 @ increment index  
    cmp r1, r2 @ keep looping?  
    bne array_sum
```

Assignment Project Exam Help

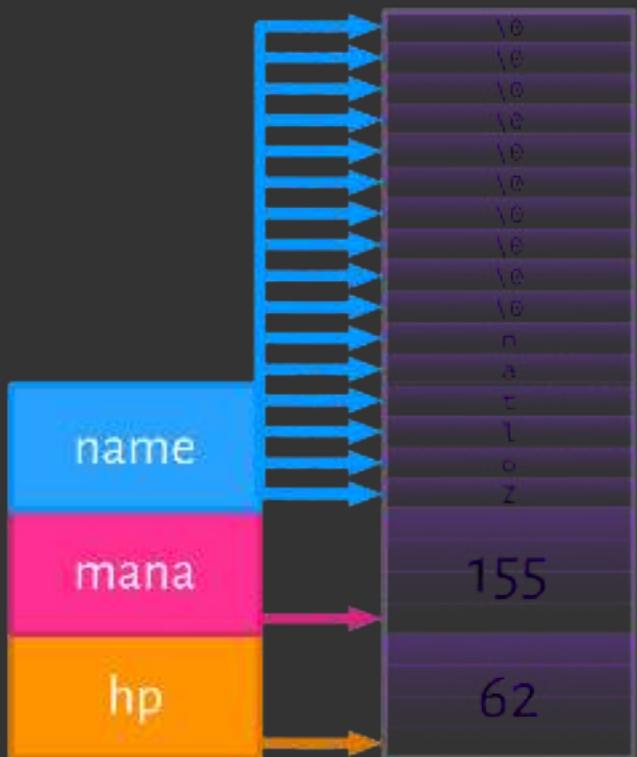
Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Records

程序代写代做 CS编程辅导



WeChat: cstutorcs

0x2000001c

Assignment Project Exam Help

0x20000018

Email: tutorcs@163.com

0x20000014

QQ: 749389476

0x20000010

<https://tutorcs.com>

0x20000008



0x20000110

0x2000010c

0x20000108

0x20000104

What's the difference between a  and a record?

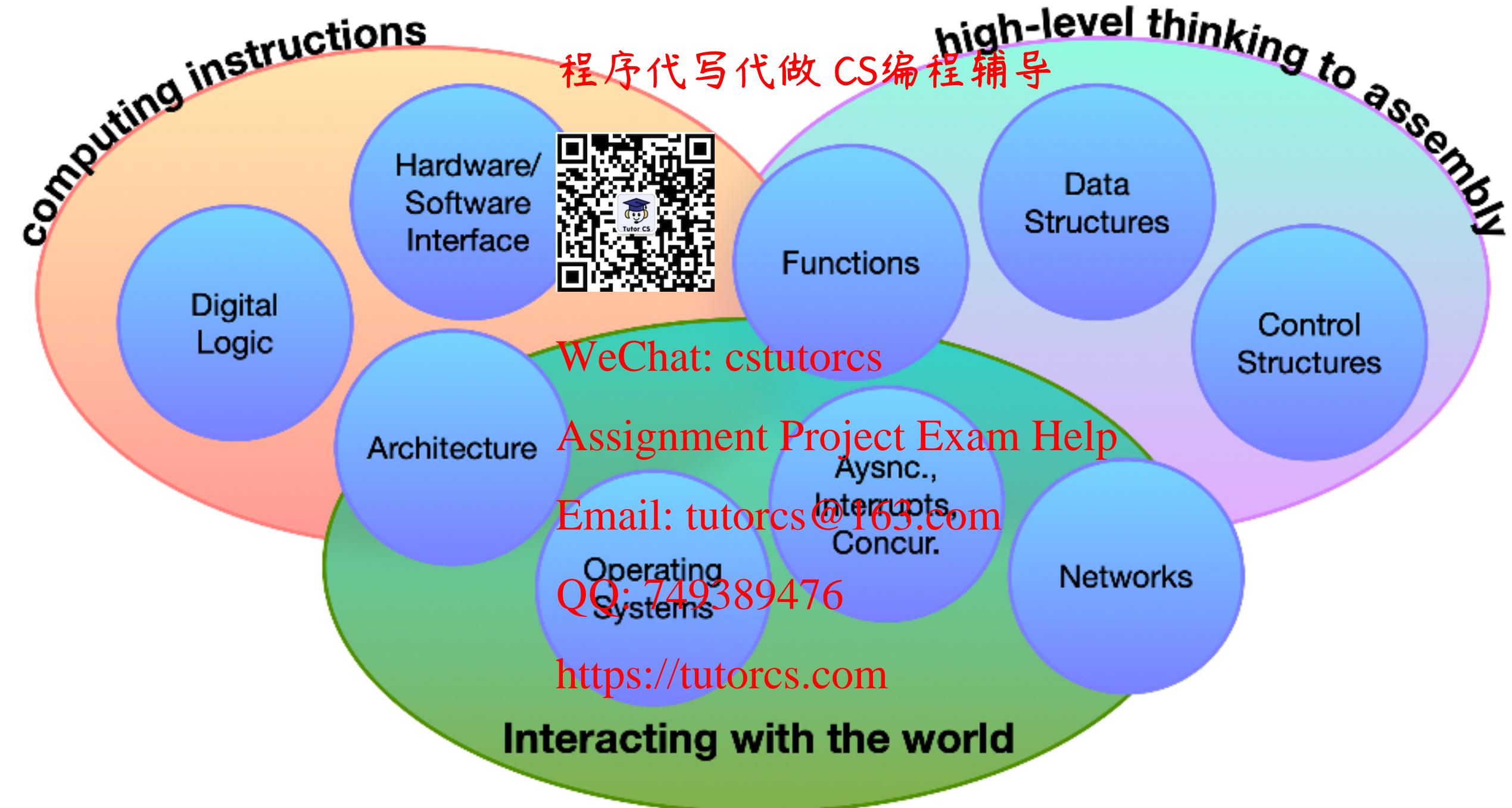
Imagine you are creating a Point-of-Sale system (cash register) using a microbit. What data structures might be required? **WeChat: cstutorcs**

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



程序代写代做 CS编程辅导



WeChat: cstutorcs

Asynchronism, Interrupts, and Concurrency

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Async Main Topics (in short)



- Interrupts & Exceptions: When
- What happens during an inter

- How is this related to **parallel computing**?

WeChat: cstutorcs

- Concurrency and Synchronisation

- Race Conditions

Assignment Project Exam Help

- Mutual Exclusion

Email: tutorcs@163.com

- Synchronisation (Locks and Semaphores)

QQ: 749389476

<https://tutorcs.com>

talk

程序代写代做 CS编程辅导

What's an interrupt? Why are they useful?



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Interrupts

程序代写代做 CS编程辅导

One or multiple lines **wired directly** to the sequencer.



Interrupts: Enables pre-emptive scheduling, timer driven actions, transient hardware interactions, etc...

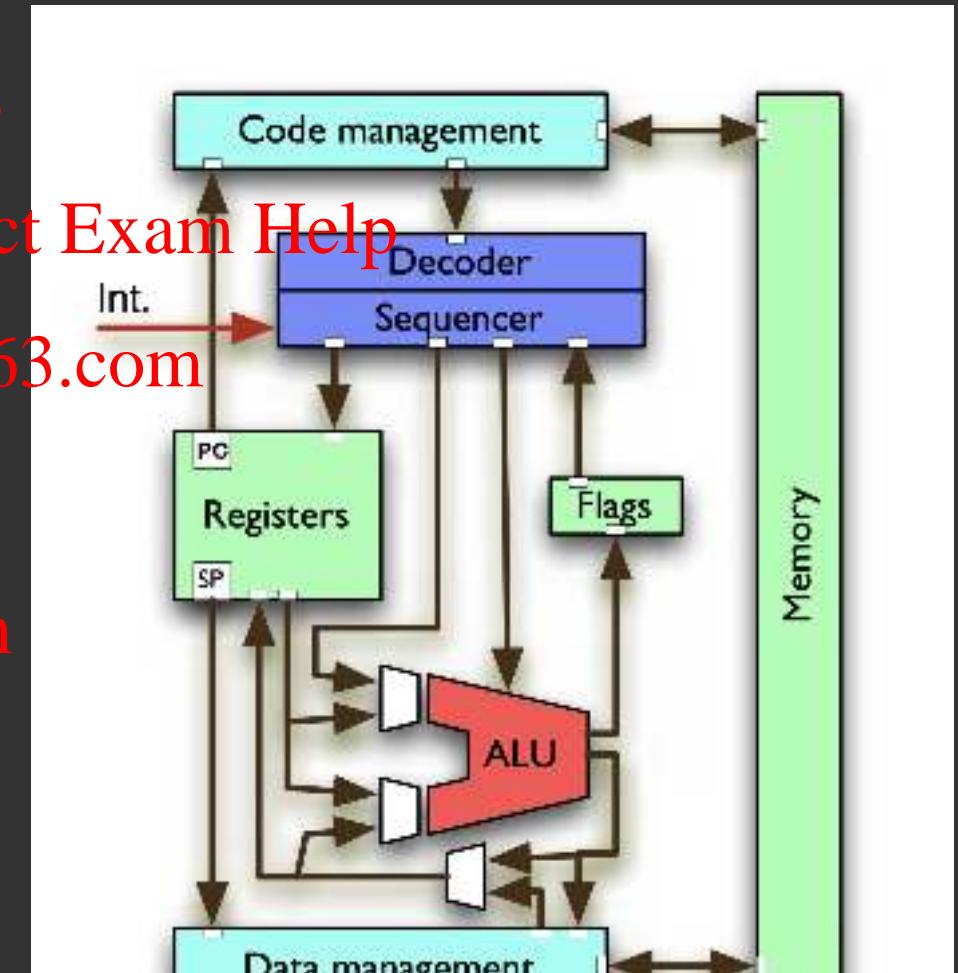
BUT: A little bit more work to set up, requires external hardware (“interrupt controller”) to encode external requests.

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Interrupt Handler

程序代写代做 CS编程辅导

A “normal function”, called when interrupt is triggered.



The CPU saves the “caller-save” context to the stack, loads `lr` with special value.

Handler saves “callee-save” context, then runs I/O or time-critical code.

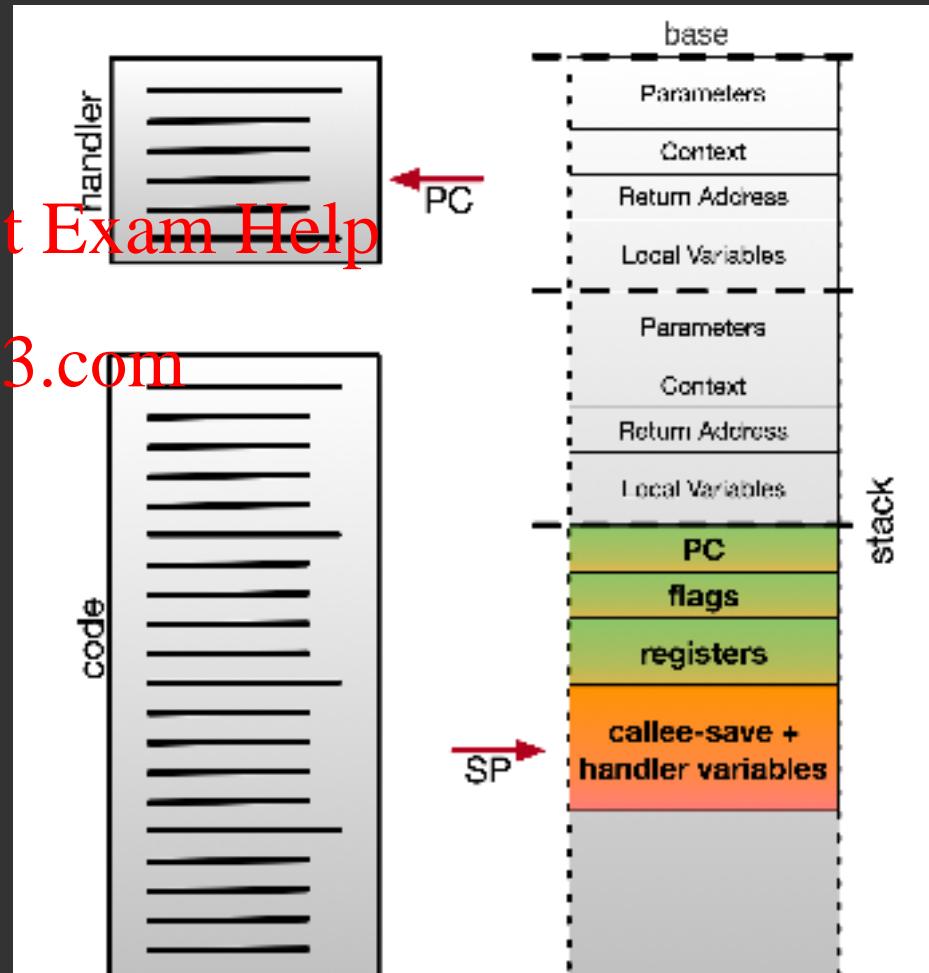
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

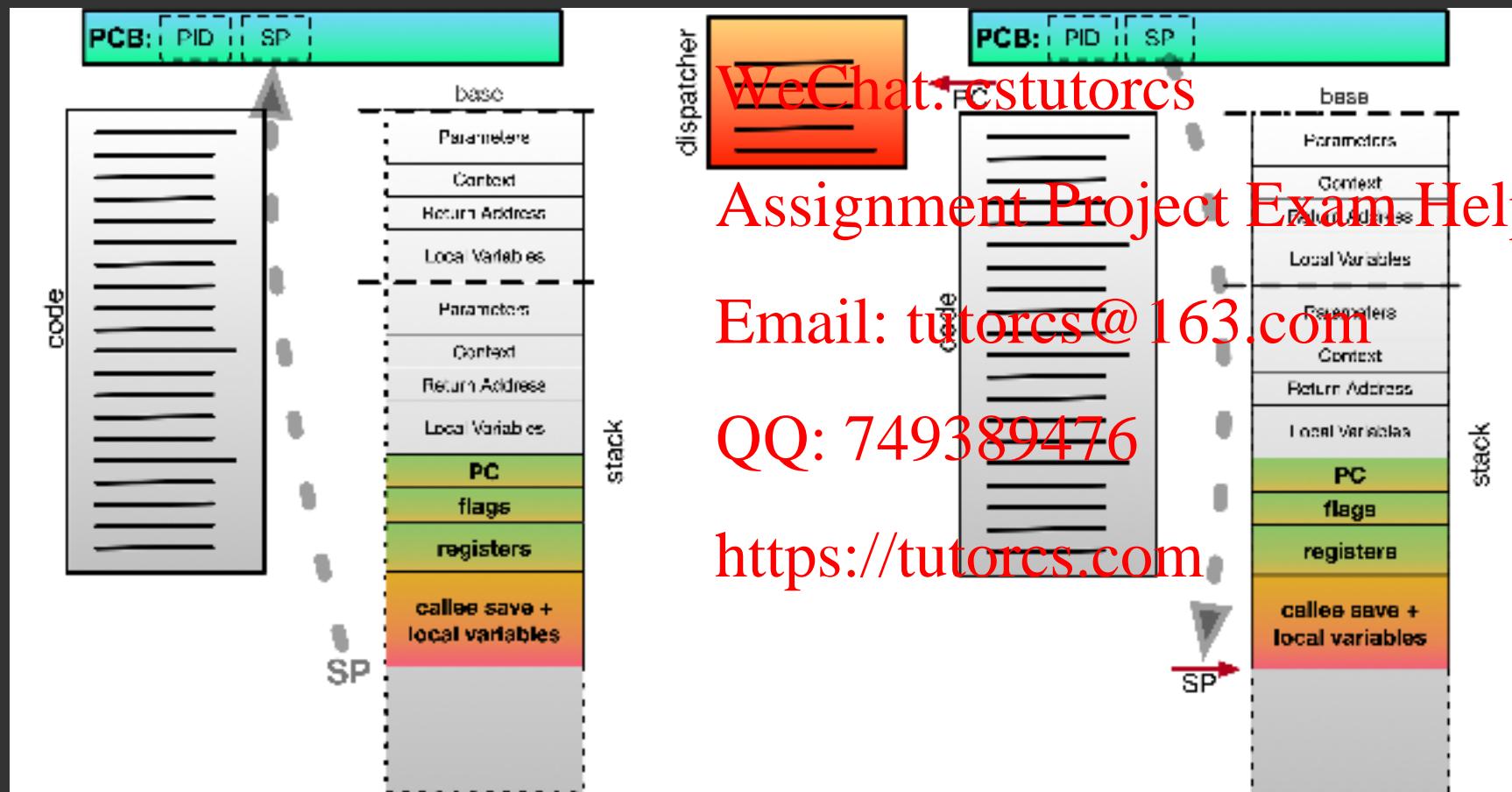
<https://tutorcs.com>



Context Switch

程序代写代做 CS编程辅导

Switch out a program (during an interrupt) without it even noticing!



What's problems can occur when writing concurrent programs? What are the possible solutions?



How does the “too much milk” problem help us understand this issue?

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Race Conditions and Mutual Exclusion



When the sequence or timing of execution has an effect on the outcome.

Can result in bugs! (e.g., in Assignment 1)

What is the value at Count in this code?

```
Count: .word 0x00000000

    ldr    r4, =Count
    mov    r1, #1
for_enter:
    cmp    r1, #100
    bgt    end_for_enter
    ldr    r2, [r4]
    add    r2, #1
    str    r2, [r4]
    add    r1, #1
    b      for_enter
end_for_enter:
```

WeChat: cstutorcs

1dr r4, =Count

Assignment Project Exam Help

for_leave:

cmp r1, #100
bgt end_for_leave

Email: tutorcs@163.com

QQ: 749389476

1dr r2, [r4]
sub r2, #1
str r2, [r4]

<https://tutorcs.com>

end_for_leave:

Critical Section

程序代写代做 CS编程辅导

```
ldr r4, =Count  
mov r1, #1  
for_enter:  
    cmp r1, #100  
    bgt end_for_enter
```



```
ldr r4, =Count  
mov r1, #1  
for_enter:  
    cmp r1, #100  
    bgt end_for_leave
```

WeChat: cstutorcs

Negotiate who goes first

Assignment Project Exam Help

```
ldr r2, [r4]  
add r2, #1  
str r2, [r4]
```

Critical section

```
ldr r2, [r4]  
sub r2, #1  
str r2, [r4]
```

Critical section

QQ: 749389476

Indicate critical section completed

```
add r1, #1  
b for_enter  
end_for_enter:
```

```
add r1, #1  
b for_leave  
end_for_leave:
```

<https://tutorcs.com>

What does ARMv7-M give us? 程序代写代做 CS编程辅导

```
Count: .word 0x00000000
```

```
ldr r4, =Count  
mov r1, #1  
for_enter:  
    cmp r1, #100  
    bgt end_for_enter  
enter_strex_fail:  
    ldrex r2, [r4] ; tag [r4] as exclusive  
    add r2, #1  
    strex r2, [r4] ; only if untouched  
    cbnz r2, enter_strex_fail  
    add r1, #1  
    b for_enter  
end_for_enter:
```



```
ldr r4, =Count  
mov r1, #1
```

for_leave:

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
<https://tutorcs.com>

```
ldrex r2, [r4] ; tag [r4] as exclusive  
sub r2, #1  
strex r2, [r4] ; only if untouched
```

```
cbnz r2, leave_strex_fail  
add r1, #1  
b for_leave
```

end_for_leave:

Using a “lock” variable

程序代写代做 CS编程辅导

```
Count: .word 0x00000000
Lock: .word 0x00000000 ; #0 means unlocked

ldr r3, =Lock
ldr r4, =Count
mov r1, #1

for_enter:
    cmp r1, #100
    bgt end_for_enter

fail_enter:
    ldrex r0, [r3]
    cbnz r0, fail_enter ; if locked
    mov r0, #1           ; lock value
    strex r0, [r3]        ; try lock
    cbnz r0, fail_enter ; if touched
    dmb                 ; sync memory

    ldr r2, [r4]
    add r2, #1           Critical section
    str r2, [r4]

    dmb                 ; sync memory
    mov r0, #0           ; unlock value
    str r0, [r3]          ; unlock

    add r1, #1
    b for_enter

end_for_enter:
```

Any context switch
needs to clear
reservations

for_

```
    cmp r1, #100
    bgt end_for_leave

fail_leave:
    ldrex r0, [r3]
    cbnz r0, fail_leave ; if locked
    mov r0, #1           ; lock value
    strex r0, [r3]        ; try lock
    cbnz r0, fail_leave ; if touched
    dmb                 ; sync memory

    ldr r2, [r4]
    sub r2, #1           Critical section
    str r2, [r4]

    dmb                 ; sync memory
    mov r0, #0           ; unlock value
    str r0, [r3]          ; unlock

    add r1, #1
    b for_leave

end_for_leave:
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

end_for_leave:

page 312 of 481 (chapter 5, “synchronization” up to page 334)

What's mutual exclusion?



Can this be achieved on a microcontroller?

How would you do it?

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Networks

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Networks Main Topics (in short)

- Transmission mediums
- Communications protocols
- Packet switched/circuit switched
- Simplex/duplex
- Topology
- Parallel/Serial
- Timing and Synchronisation
- OSI reference model (7-layers!)



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

How many transmission medium
name?



If you were stuck on a desert island what transmission media could you use to send a message for help?

WeChat: cstutorcs

What is a communications protocol?

Assignment Project Exam Help

Why would it be needed?

Email: tutorcs@163.com

Explain your answer.

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导

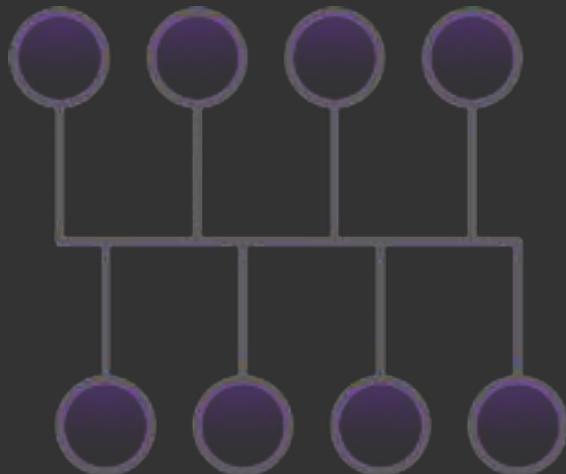
Topology

topology is the way that the nodes are connected to one another (both physically and logically)



there are several different ways to connect the nodes together, each with pros and cons

WeChat: cstutorcs



bus



ring



star

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Serial vs parallel

程序代写代做 CS 编程辅导

serial

data is sent one-bit-at-a-time

fewer bits sent per signal, but si



parallel

multiple bits sent simultaneously (e.g. multiple wires)

to keep all the connections in sync



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Synchronous vs Asynchronous

程序代写代做 CS编程辅导

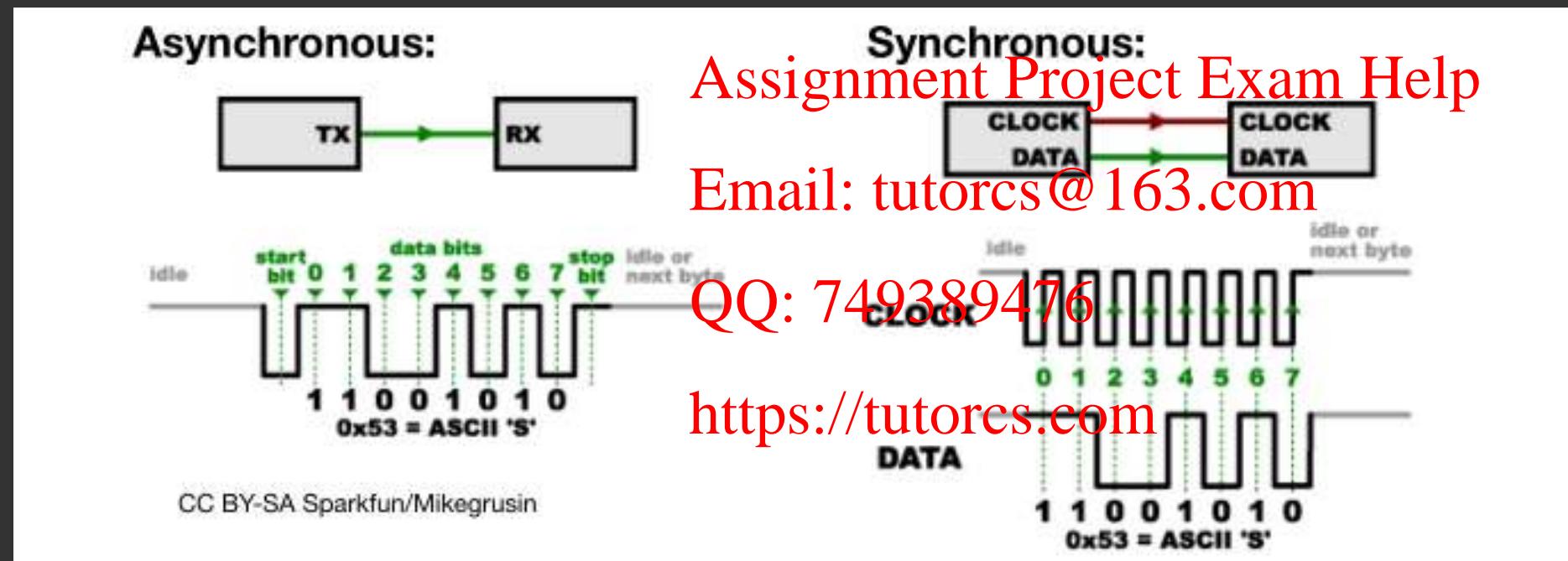
synchronous
transitions on a *clock line*
no clock skew issues, but require
extra connection



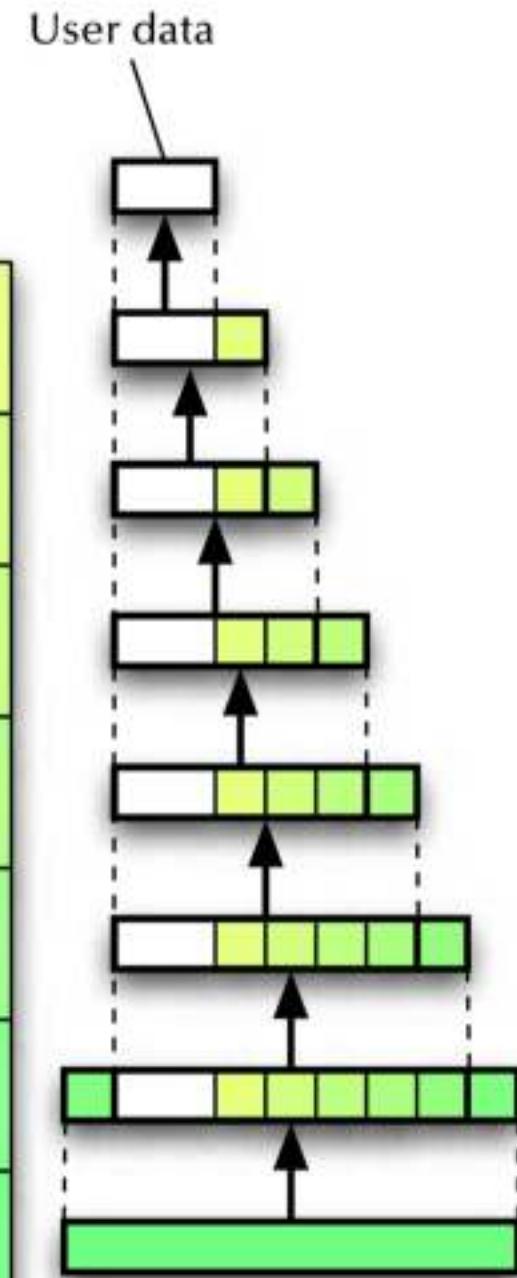
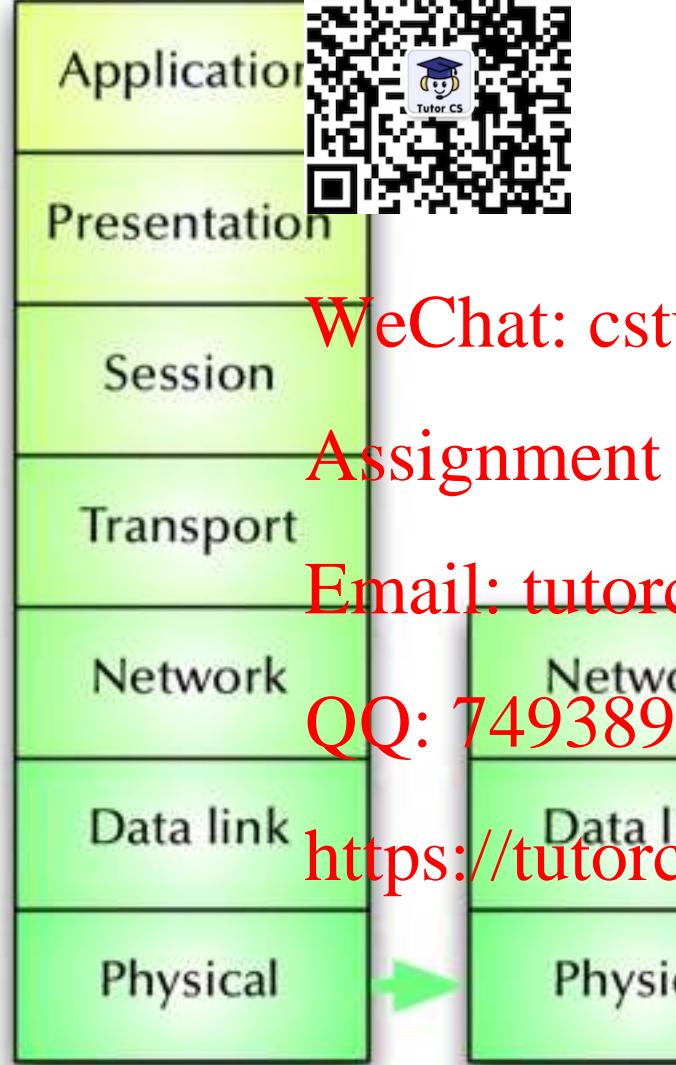
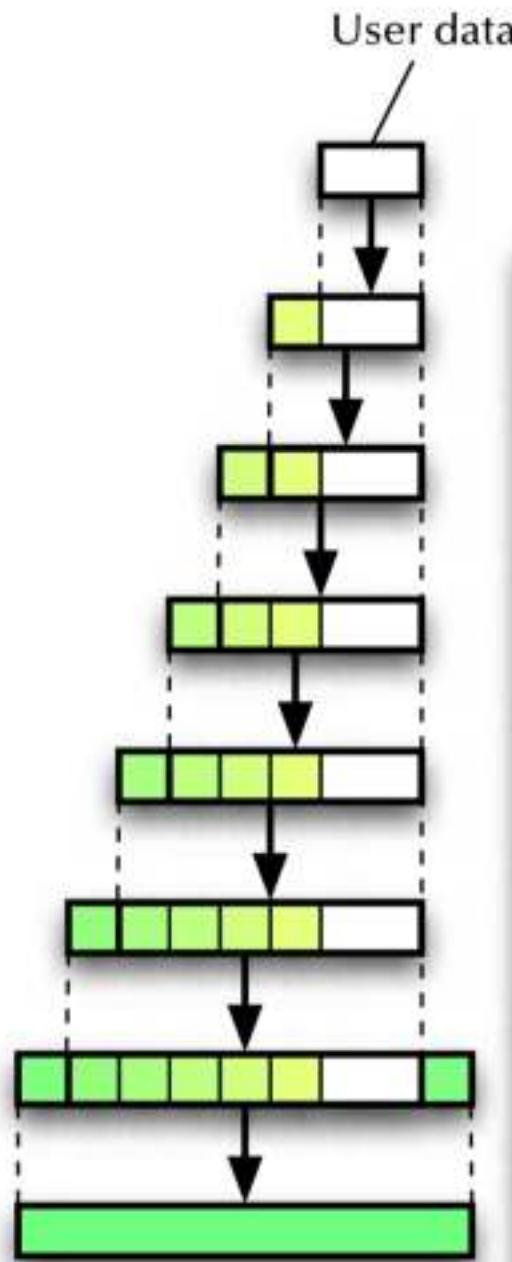
chronous
pendent) timers at each end

extra connections required, but more vulnerable to

synchronisation issues
WeChat: cstutorcs



OSI Network Layers



程序代写代做 CS编程辅导



WeChat: cstutorcs

Operating Systems

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

OS Main Topics (in short)

程序代写代做 CS 编程辅导

- Operating Systems: Concept
- OS Categories



- OS Architectures

WeChat: cstutorcs

- Processes - what are processes anyway?

Assignment Project Exam Help

- How do OSs handle processes?
- Scheduling

Email: tutorcs@163.com

Why do we need operating systems
QQ: 749389476

<https://tutorcs.com>

What's an OS? ...two main roles

virtual machine

provides friendly & safe environment

memory management

hardware abstraction

process management

inter-process comms (IPC)



resource manager

coordinates access to resources

processors

memory

WeChat: cstutorcs

mass storage

communications channels

Assignment Project Exam Help

devices (timers, GPUs, DSPs, peripherals...)

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导

Kernel: definition

the **kernel** is the program (functionality, data structures in memory, etc.) which performs the *core* role(s) of the OS



access to the CPU, memory, peripherals all happens *through* the kernel through a **system call**

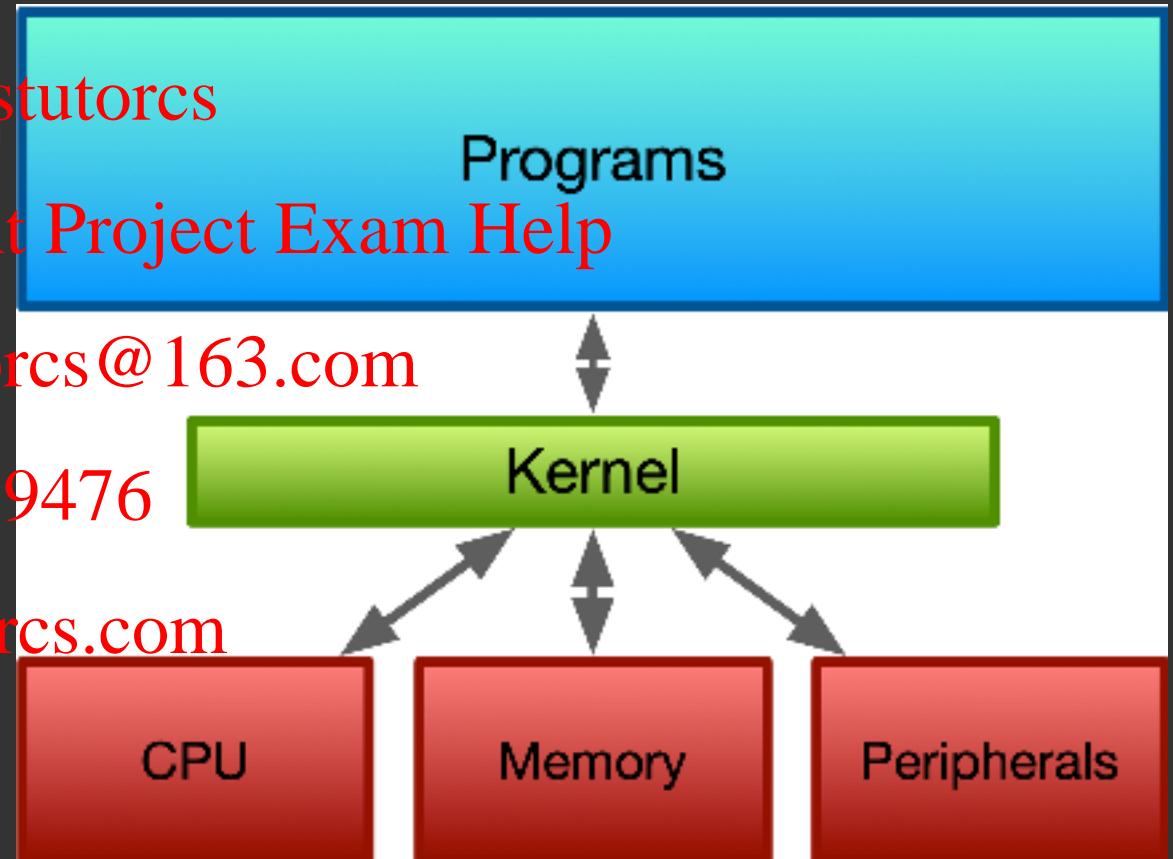
WeChat: cstutorcs

Assignment Project Exam Help

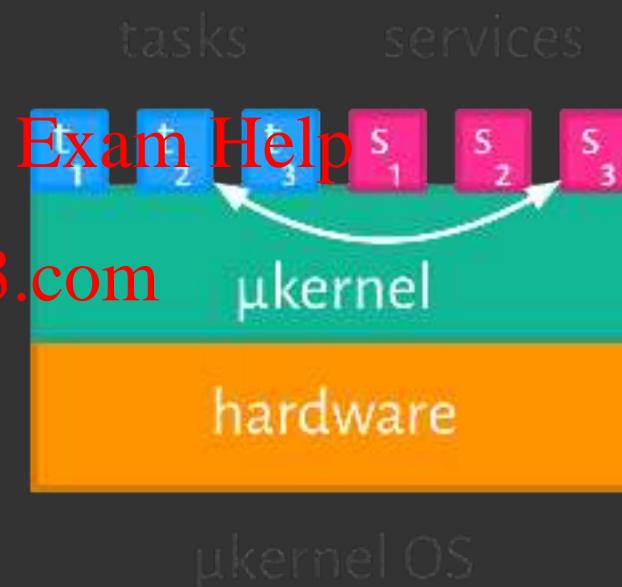
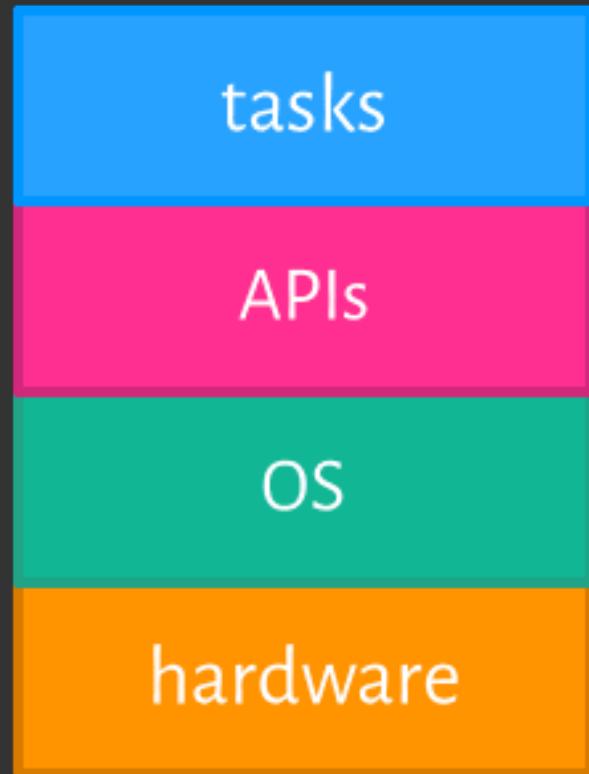
Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Monolithic OS - Modular - μ Kernels



monolithic OS

monolithic &

Why are kernels and (user) programs separate?



How are they different?

Aren't they both programs?

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Process: definition

- a *running* program
- includes the code (instructions) of the program, and the current state/context:
- registers/flags
- memory (stack and heap)
- permissions/privileges
- other resources (e.g. global variables, open files & network connections, address space mappings)

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutors@163.com

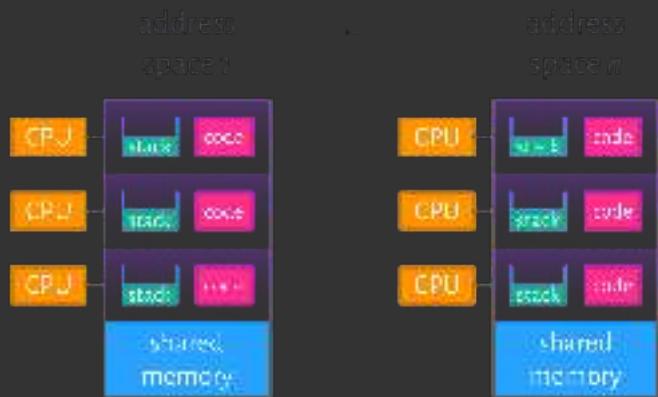
QQ: 749389476

<https://tutorcs.com>



process control
block(s)

Mapping processes to CPUs



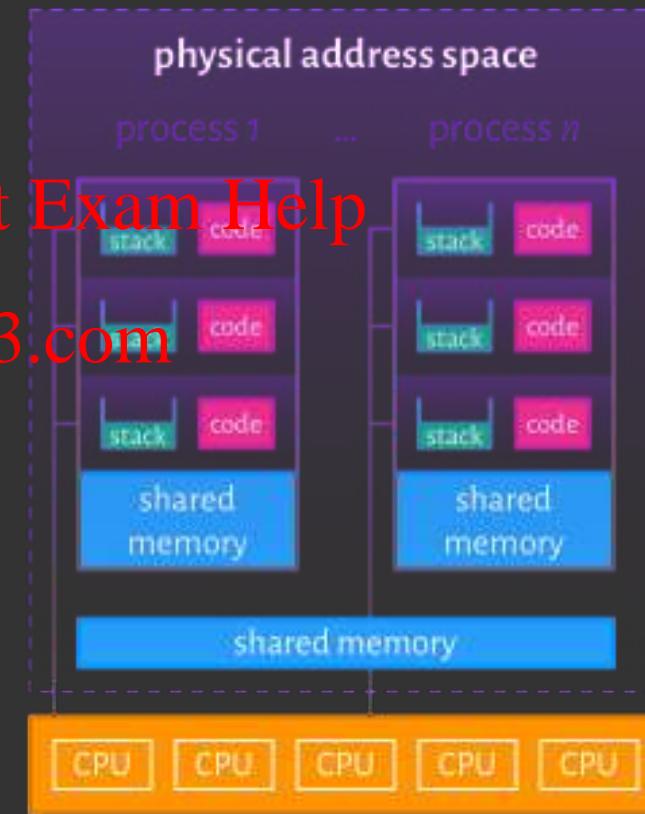
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



Architecture

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

Architecture Main Topics (in short)

- History of computing architecture
- Harvard vs von Neumann architecture
- Pipelines
- Out-of-order execution
- Vector/SIMD instructions
- Hyper-threading
- Multi-core computing
- Virtual Memory
- Alternative architectures (Parallax Propeller)



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

A simple CPU

程序代写代做 CS编程辅导

- **decoder/sequencer** converts instruction control signals
- **arithmetic logic unit (ALU)** performs mathematical operations
- **registers** provide small, fast storage to the CPU
- **flags** indicate the states of the latest calculations
- **code/data management** for loading/storing, caching
- **memory**



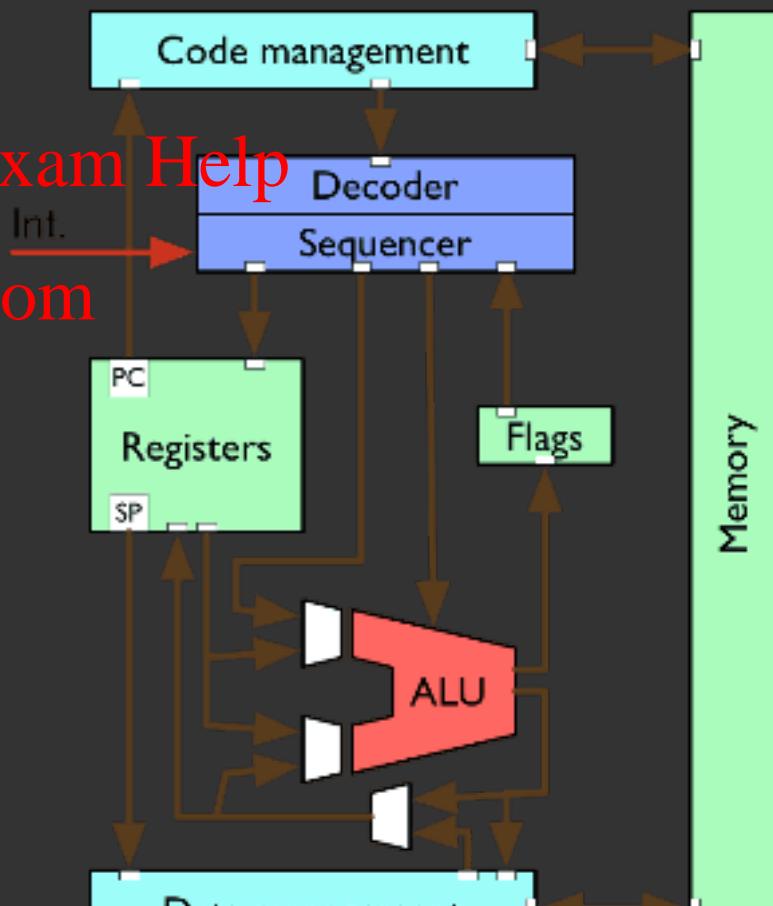
WeChat: cstutorcs

Assignment Project Exam Help
Email: tutorcs@163.com

Are any of these components unnecessary?

QQ: 749389476

<https://tutorcs.com>



Simple Pipeline

程序代写代做 CS编程辅导



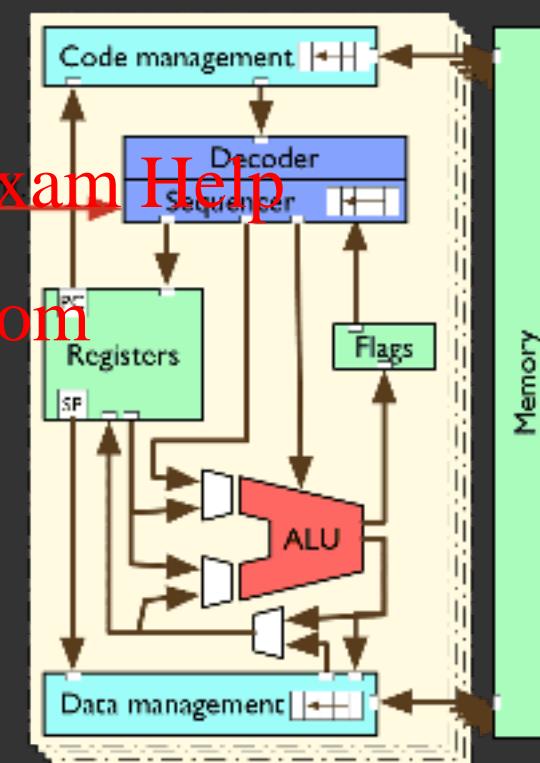
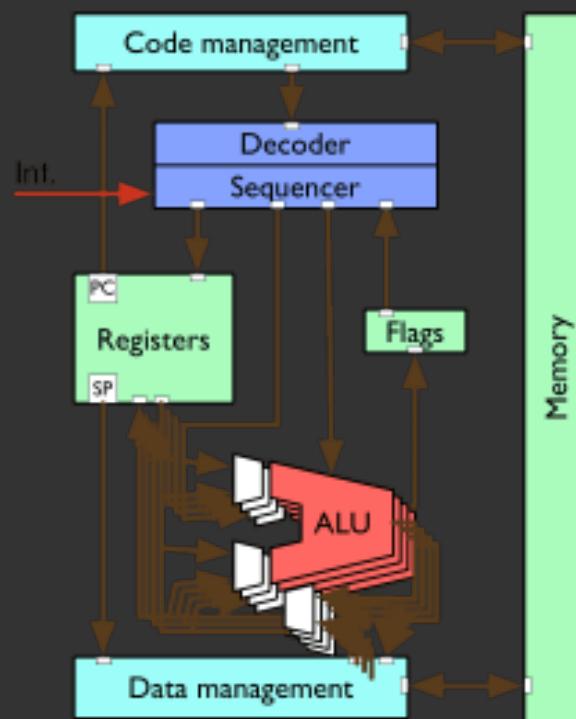
Clock Cycles
WeChat: cstutorcs
Assignment Project Exam Help

Instructions	1	2	3	4	5
1	Fetch	Decode	Execute		
2		Fetch	Decode	Execute	
3			Fetch	Decode	Execute

What can go wrong in a pipeline?

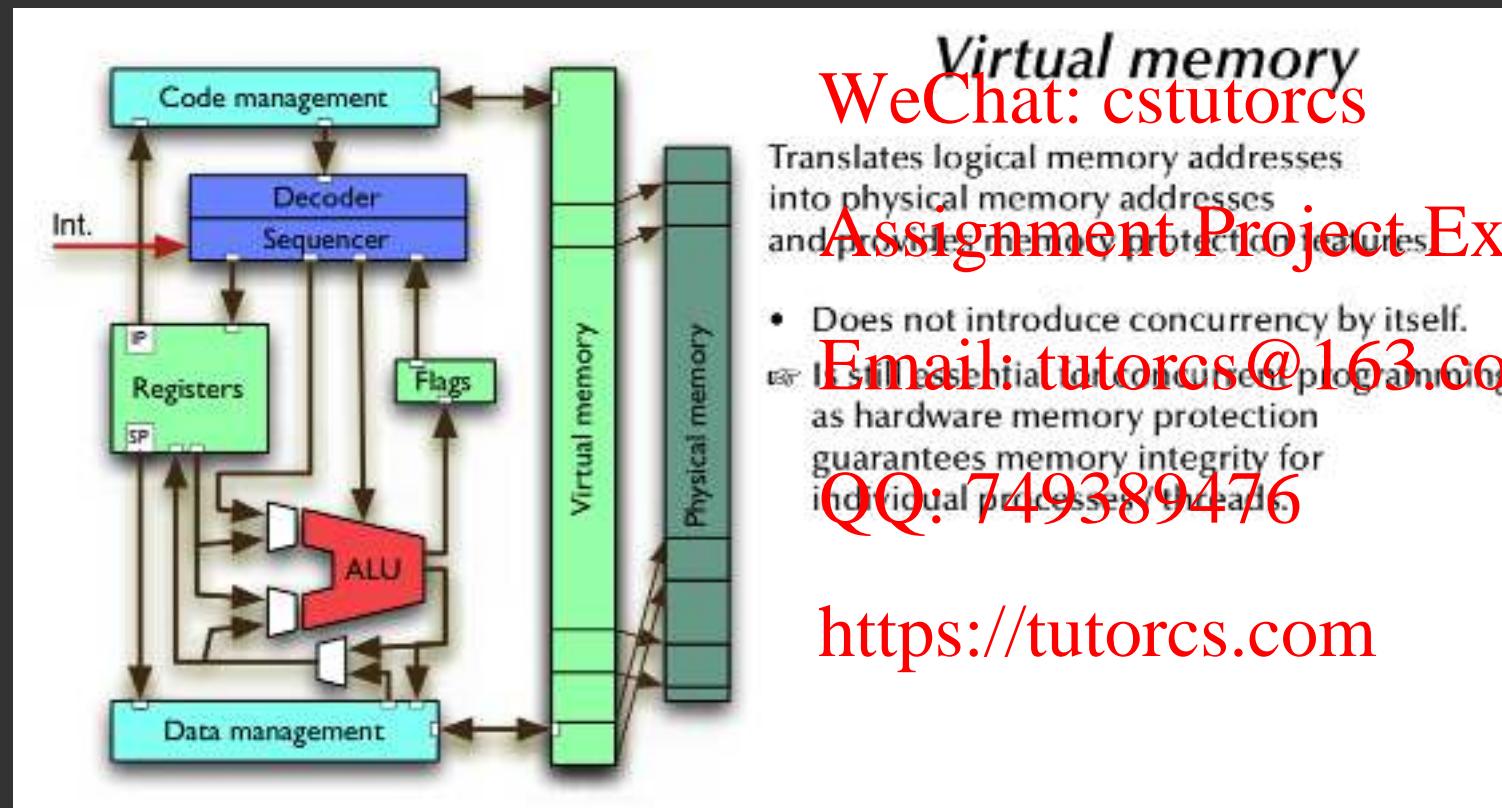
Vector/SIMD vs hyper-threading vs multi-core

What are the differences?



Virtual Memory

程序代写代做 CS编程辅导



Virtual memory
WeChat: cstutorcs

Translates logical memory addresses
into physical memory addresses
and provides memory protection features.

- Does not introduce concurrency by itself.

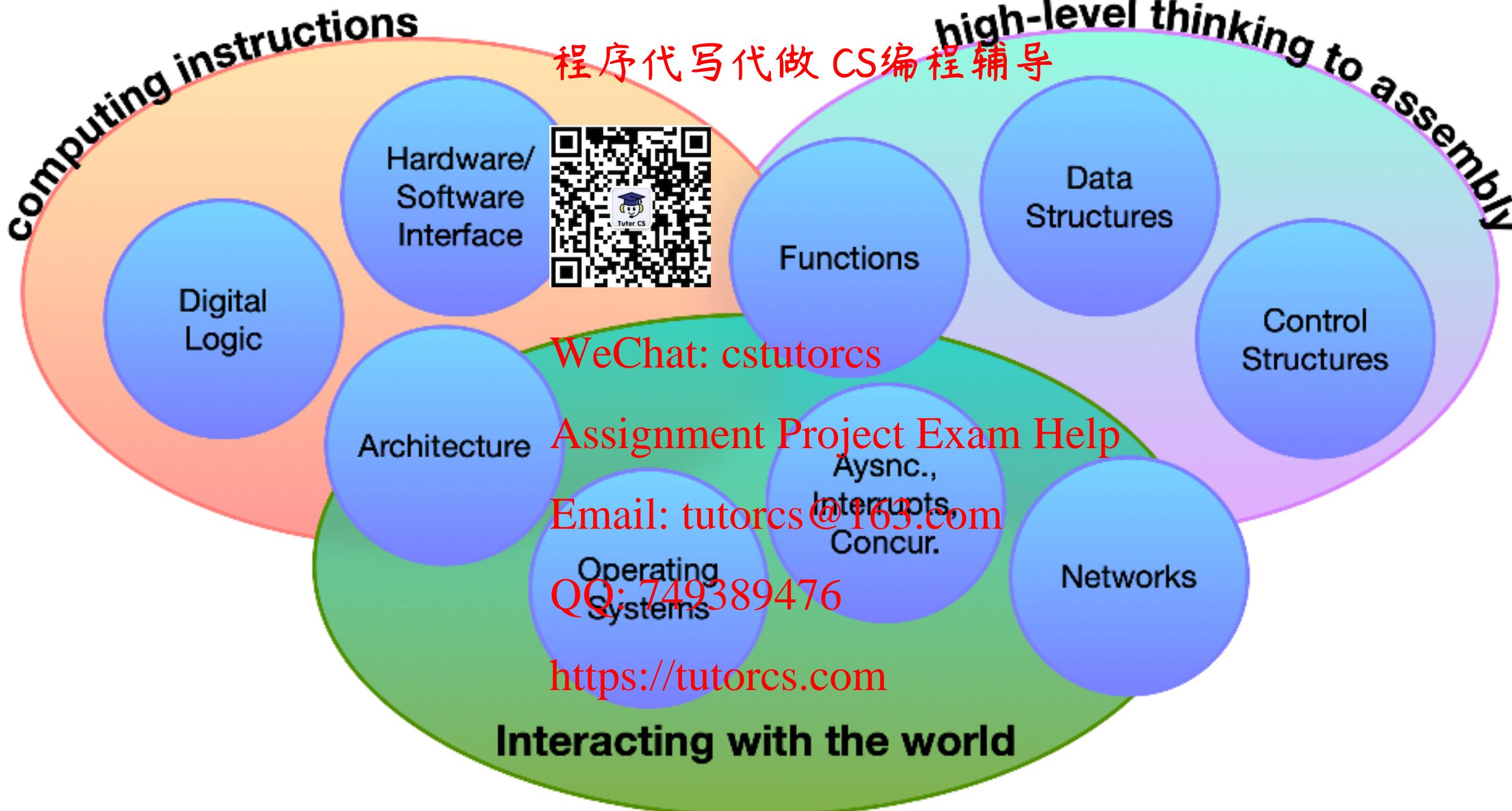
☞ It is essential for concurrent programming

as hardware memory protection
guarantees memory integrity for
individual processes/threads.

Email: tutorcs@163.com
QQ: 749389476

<https://tutorcs.com>

Is virtual memory an architecture (hardware) topic or an OS (software) topic?



程序代写代做 CS编程辅导



Finally done. That was epic, thanks for
coming with me everybody!
WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>