2016-6-14 16:37:16 【GR6018A】 I2C模块

ADT7410读ID失败，去掉R237、R238（*硬件连接：ARM\_SDA ->R237 ->EX\_SDA->PCA9306DCTT 模块*）后，读ID正常。（另外，实际抓到的读时序和手册标准读时序有区别：发送设备地址和内部寄存器地址之间存在一段较长的低电平（SCL），类似于重新开始信号，暂不知道是什么原因）

两次发送之间的低电平为MCU处理第二次发送操作的时间

2016-6-15 09:54:56 【GR6018A】 CAN未响应(ack)

CAN0、1收发OK。原来的问题是CAN0能发送，总线上发送的波形也有，但是CAN1未响应，现将CAN模块通讯速度提高1倍后（分频系数由8->4），CAN0、1之间能正常收发了。

2016-6-15 14:30:59 【GR6018A】 XEBI模块

CLPD上连了两个CS线，意味着CS1(0x20\_0000 – 0x3f\_ffff)到CS0(0x40\_0000-0x7F\_FFFF)这段空间用来访问CLPD。

2016-6-16 09:26:24 【GR6018A】 时钟模块

外时钟模式下，无法进行烧写。(原因呢？)

PS: 电阻焊接R152 R150 R140 R148 型号：1020封装 3m欧

* + 1. 0:32:03 【GR6018A】

1.XEBI模块

EBI时钟输出已经有ECLK，另一个ECLKX2又是用来做什么的?(目前配置是ECLK使能，ECLKX2关闭)。看手册上说，这个时钟twice of bus clock，难道这个时钟的作用是提供更快的访问？)

ECLKX2硬件未连接

nEWAIT信号没看懂怎么用。（猜测：短时间内，MCU进行多次外部访问，但外部设备处理不过来，为了保证所有访问正常执行，外部设备向MCU发送等待信号，保证每次访问的同步执行。MCU检测到nEWIT信号有效后，停止当前发送，插入等待周期（大小由EXSTR0和EXSTR1决定））

实际应用，如果xebi外挂的是FPGA，可由FPGA来给出，保证其操作完整运行；如果是外挂flash，则由flash器件本身给出该信号。

2.MCU模式选择

模式切换是否成功，如果切换成功，为什么访问不了CS2片选空间。

2016-6-20 10:58:09 【GR6407】 double取整

Double Ceil(double inputValue) (四舍五入)

Double强转int不会丢掉数据吗？(强转舍去小数部分)

2016-6-22 09:23:40

【GR6018A】

1. CS2映射到那一块本地地址？
2. 如果没有映射，如何映射？
3. 和prm文件有什么关联？

【GR6407】

1. 接收速度达到7 MB/S左右时，接收线程（或处理线程）的调度优先级很高，导致UI线程得不到调度而长时间卡顿，甚至卡死；
2. 确认是CVI的线程池机制问题，还是操作系统的调度算法问题？
3. 还有个问题是接收速度如何提升到20MB（最少）？

CVI事件驱动函数都保存在\*.uir文件中，每个控件名和函数关联（具体：在uir文件可以看到，事件驱动函数和它的控件名紧靠在一起），那么事件驱动函数注册流程猜测为：

Loadpanel:将控件名和函数对读到内存中——》创建驱动事件链表(或者数据)（成员：控件名+函数）——》将事件驱动插入链表（保存在数组）。

问题：控件事件如何与注册的函数关联

**2016-6-25 11:14:38 【GR6407】**

1. 启动接收、处理线程后，UI卡顿很明显；（UI卡顿是因为主线程得不到调度，或者调度机会少，导致UI卡顿；改进：线程池创建放在初始化的地方，线程调度不再关注ID，即移除线程清除处理）
2. 在拆分I、Q数据时，循环次数太长，导致处理线程执行效率低;（有没有办法以追加的方式一段段的刷新）
3. 点击启动按钮时，UI也会卡顿（将线程池创建放在EXE初始化处，有改进；）
4. 程序停止接收数据后，（对方还在不停发送数据）如果不清除本次连接，CPU占用率将会很高，进而导致UI线程无法及时响应。

**2016-6-27 15:08:06 【GR6407】TCP buffer生命周期**

1. 多次创建和断开TCP，无法收到正确的数据，每次校验和都是错的。（如果一直保持TCP连接，不会出现该问题）

原因：一直保持TCP连接，为该连接分配的缓存是不会销毁的，自然，缓存中待读数据的大小也不会变化，当客户端下次重新启动任务，将能接着上次状态继续读缓存数据，而不会导致数据链路中断。

但是，如果客户端每次停止任务时关闭TCP连接，那么本次连接的缓存也会被销毁，自然，缓存中的数据也跟着消失。

**2016-6-30 09:28:58 【GR6018A】函数参数入栈**

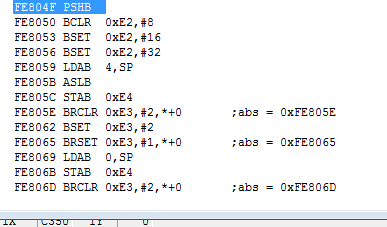
MC9S12X 函数调用的汇编过程：

LDAB #72 ;addr1 入栈

PSHB

LDAB #11 ;data 入栈

CALL 0x804F,0xFE



1.函数参数入栈顺序是编译器特性，与操作系统无直接关系。源码编译成汇编代码，汇编代码会体现函数参数入栈的顺序。如上图所示，函数参数是从右往左入栈。

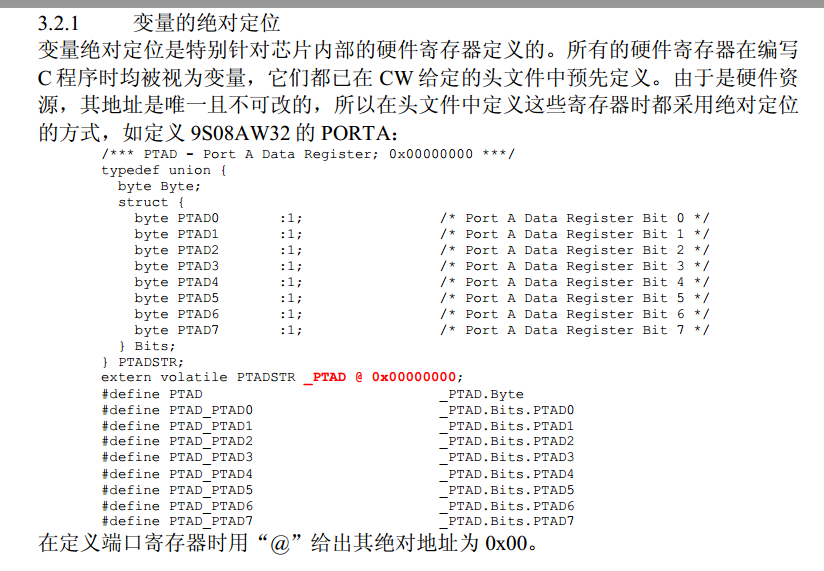
2.CALL指令的作用：将函数调用的下一条指令地址压栈（SP -= 2;SP -= 1），函数调用完后再将该地址弹到PC寄存器（RTC），以便自动执行下一条指令。

**2016-7-15 10:01:41 【GR6018A】xebi**

Xebi读数据片选时间过短，导致MCU没有读到FPGA出的数据。解决办法：通过配置EBICTL1寄存器来延长CS的时间。

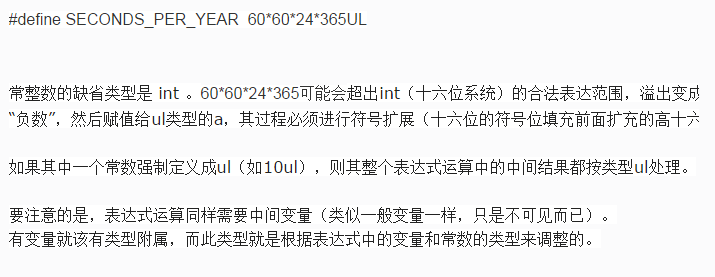
**2016-7-18 16:21:21 MC9S12XEQ512MAA 单片机prm文件学习**

1. extern volatile DDRKSTR \_DDRK @(REG\_BASE + 0x00000033UL);含义



编译器特性？

1. 0x00000033UL 意义



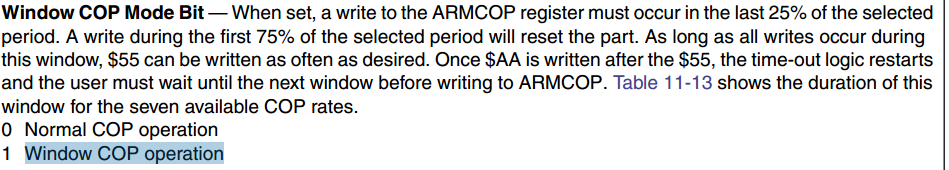
常数的强转符号

Ul在32位机上的字节数，在64位机器上的字节数？

1. Window COP operation.窗口看门狗

窗口：形容某事发生的最佳时段。

窗口看门狗使能位说明：



可以看出，窗口看门狗实际上就是指**高精度**的复位看门狗。

**2016-7-25 14:41:46 ADT7410**

1. 过温、欠温的应用以及故障队列

过温：高低温报警

欠温：监视环境是否处于某一特定温度

故障队列：防止误触发。当过温或者欠温连续发生若干次（软件可编程：最大4）时，才激活INT或者CT引脚。

**2016-7-25 16:21:02 【GR6018A】远程加载**

1. mcs文件总字节是4236233，最后两个字节是OD OA（在UE中以16进制打开 == 回车、换行？）。经过fread读到内存后，末尾两个字节变成：0A 00。

Mcs打开方式fopen: r+，即以ASCII方式打开读写，当读到回车换行时，自动将文本中0x0d 0x0a转换成0x0a 0x00。

**2016-7-26 11:56:09 【GR6018A】norflash烧写**

1. MCU从“地址0”（由CPLD减掉片选空间基地址）连续写入FPGA的mcs文件（大小4M左右）后，再从“地址0”读回，读回内容与原mcs文件一致。但是由CPLD控制FPGA从该块FLASH启动时，启动失败。

通过jtag直接将Mcs文件烧到flash中，再用同样的方式启动，启动成功。Jtag烧写的地址不确定，CPLD控制FPGA启动时读flash地址确定，另外，假设jtag和CPLD烧写与读取都是从flash地址0开始，是否怀疑MCU写的地址0与实际flash的地址0不是对应关系？

确认MCU是从norflash的地址0开始连续写入。

1. 将Jtag写入flash中的mcs文件读回，与原mcs文件作比较，发现最后几个字节不一致。

Mcs文件是iMpact生成的格式化文件，上位机需要去掉格式，提取有效数据。关于Mcs文件的格式说明：

<https://forums.xilinx.com/t5/Welcome-Join/Required-mcs-data-format-information-for-FPGA-configuration/td-p/228293>

**2016-7-28 11:04:18 【GR6018A】遗留问题整理**

1. **prm文件**中关于各类段与实际物理内存的对应关系，以及将变量定位特定段的方法（#pragma）；字符串、全局变量，静态变量保存的各个段

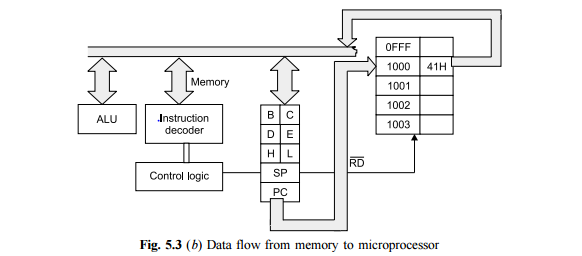
#pragma 预处理指令，标准C扩展语法

1. **MC9S12从上电到main函数的过程，以及时间**

上电首先工作的是复位电路，复位持续时间内，时钟信号是锁存的，即各模块无时钟输入（所以说首先工作的是复位电路），待VCC和时钟稳定后，释放复位信号，时钟模块输出脉冲。CPU依据稳定后的内部振荡时钟开始运行，运行地址由PC寄存器决定（一般是0000H）。

**CPU执行指令过程**

取指---译码---执行



以8085CPU为例，假设下一条要执行的指令是: MOV B,05H //B CPU内部寄存器

（MOV指令的机器码：41H）

Machine cycle = n CLK period

取值：如图所示，control单元将pc中的地址1000H输出到Address bus上。具体控制：

S0

Memory

Control unit

S1

IO / M

RD

ALE

1）IO/M拉低，选中访问对象为memory，S0、S1拉高，即s1=s0=1,表示接下来操作是取指令。

2）由于8085的低A位地址和数据总线分时复用，所以PC寄存器将地址高8位，即10H送到地址总线，将低8位00H送到数据总线，ALE拉高，表示低8位传输的是地址而不是数据。

3）地址送出的同时（或者之后），Control Unit将RD读Memory使能信号拉低，Memory将地址1000H单元的数据41H送到数据bus上。（CPU 的RD信号对应的是Memory的OE信号，即数据输出使能）

4）41H进入CPU后，先存入DR，然后再存入IR，经过decoder译码后，选择控制单元进行读外部memory操作

5）读外部memory与取指令类似，IO/M拉低，表示访问Memory，S0拉低、S1拉高表示接下来的操作是read data(not fetch opcode)，然后就是给除16位地址（H、L），给RD读使能信号，接收memory送出的数据，保存在B寄存器中。至此，一条指令的执行完毕。

CPU的控制单元的控制体现在时序图上，硬件控制体现在：选择对应的电路（如加法、减法），输出控制信号（如S1 S0，ALE，RD等）

遗留问题：

1. 控制单元如何控制PC寄存器输出地址，何时输出？（控制信号输出前，还是控制信号输出后，由谁控制？）

不一定要控制，当CLK上升沿到来时，PC寄存器的值直接输出，以后都保持前一个输出的状态，但是否有效由其他控制信号决定，比如ALE信号控制，当ALE为高时，允许地址信号输出到AddressBus上，否则地址/数据信号引脚状态不变，即PC输出的地址信号无效。

1. CPU开始工作后，最先工作的unit是哪一块（ALU、控制、PC寄存器）？

CPU总是从读指令开始，那么，可以大胆猜想设计CPU时，将控制单元的初始状态配置成读指令（指令执行完成后也恢复到初态），具体到电路上就是通过开关、放大、增益等手段，将控制信号（S1、S0、IO/M等）的初始输出信号（电压）与引脚XX相连，并将其设置成对应的1或者0电平。

这样一来，当时钟稳定输出后，CPU就开始依据预设的状态，开始输出控制信号和地址信号，Memory接收到对应信号后，将第一条指令送到CPU。

1. fetch opcode时，控制信号和地址（数据）信号的时序逻辑关系，以及和CLK之间的关系？

假设，在时钟上升沿阶段，地址信号先送到Addrbus上，控制信号随后再输出（同一个时钟上升沿），那存在的问题就是，地址信号先到达Memory，如果Memory不做处理，这些信号能否保持？

取决于Memory地址/数据端口内部电路实现。

1. 信号的理解

电压，电子，电势能

1. Memory对于地址的处理

以AT89S51为例，当其端口（如P0）为地址/数据输入时，引脚P0.x通过一个三态的数据输入缓冲器直接接入到内部总线。所以，Memory是否能处理CPU送出的地址信号，实际上取决于Memory端口的电路结构，更具体的讲，就是Memory的端口电路是否设置的有限制地址/数据信号输入的门电路，比如，设置一个由三极管控制的开关，将IO/M信号作为三极管的一端输入，当IO/M为高时，与地址/数据信号相关的输入端截止，即输入的信号此时不会改变三极管的输出，当IO/M为低时，则恰恰相反，内部的三极管输出端会随着端口上输入的信号变化而变化，即AddressBus输入的地址/数据信号有效。

3）门电路：与或非，锁存器

1. 指令结束标志？

指令可以没有结束标志，当一个指令执行完后，控制信号回到初始状态，CPU则又开始从取指令运行，也相当于上一条指令执行结束。

*题外话：*

*什么是理解：明白是什么。要明白理解的概念（其他概念的理解也是一样的），多接触实例，通过现象看本质，对于“理解”来讲，现象就是理解的方法、内容和对象；*

*思维方法很重要：顺着想，反向证明，对比，联想，递归，穷举，*

*类比数据证明法：比较法、综合法、分析法、数学归纳法、反证法*

***比较法****：作差比较，a-b<0 --> a<b*

***综合法****：由已知推向结论。与分析法相反，综合法从题目给予的条件开始推导，如开始条件a不等于b的，推出(a-b)2>0，如此一步一步推，最后可以证明a3+b3>a2b+ab2。*

***分析法****：从结论寻找已知。如证明a3+b3>a2b+ab2，通过因式变化，最后得到要求证明（a-b）2>0，要是前式成立，则要求条件是a不等于b，刚好题目要求满足。*

***数学归纳法****：通常被用于证明某个给定命题在整个（或者局部）自然数范围内成立。最简单和常见的数学归纳法是证明当n等于任意一个自然数时某命题成立。证明分下面两步：*

*证明当n= 1时命题成立。*

*假设n=m时命题成立，那么可以推导出在n=m+1时命题也成立。（m代表任意自然数）*

***反证法：****是一种论证方式，它首先假设某命题不成立（即在原命题的题设下，结论不成立），然后推理出明显矛盾的结果，从而下结论说假设不成立，原命题得证。反证法的逻辑原理是逆否命题和原命题的真假性相同*

*证明：*http://e.hiphotos.baidu.com/baike/s%3D21/sign=d0ff5f1c3e292df593c3ab14bc313340/aa64034f78f0f736ebd2d6320e55b319ebc413d0.jpg *是无理数。*

*假设命题不真，则*http://e.hiphotos.baidu.com/baike/s%3D21/sign=d0ff5f1c3e292df593c3ab14bc313340/aa64034f78f0f736ebd2d6320e55b319ebc413d0.jpg *为有理数，设* http://a.hiphotos.baidu.com/baike/s%3D54/sign=8337f10a4f36acaf5de096f87dd9d48e/8694a4c27d1ed21b462c9cd6a96eddc451da3f1f.jpg*即最简分数的形式。*

*则*  http://e.hiphotos.baidu.com/baike/s%3D51/sign=fd6d09c83ddbb6fd215be52708244fea/54fbb2fb43166d22e3fda8a7422309f79052d27f.jpg , http://a.hiphotos.baidu.com/baike/s%3D63/sign=cf6678ddb08f8c54e7d3c62c3a293768/ac4bd11373f082023a982f0c4ffbfbedab641bfe.jpg

*所以* http://a.hiphotos.baidu.com/baike/s%3D16/sign=61a1c193f203738dda4a0824b21ba4db/b3119313b07eca80bd9887d4952397dda1448360.jpg *为偶数，则*  http://c.hiphotos.baidu.com/baike/s%3D9/sign=2c4983b00b2442a7aa0ef195d1196f/730e0cf3d7ca7bcbb85e7e1cba096b63f624a8d8.jpg *为偶数，可表示为*  http://e.hiphotos.baidu.com/baike/s%3D17/sign=10e593385fb5c9ea66f307e4d539f335/960a304e251f95caf5b63785cd177f3e67095245.jpg

*则*  http://c.hiphotos.baidu.com/baike/s%3D107/sign=c479c365d4160924d825a61be307359b/4b90f603738da9774b962218b451f8198618e34c.jpg

*所以*  http://g.hiphotos.baidu.com/baike/s%3D64/sign=e42ef3daf6d3572c62e29fd88b1383b6/55e736d12f2eb93875afdec1d1628535e4dd6fc7.jpg

*则* http://h.hiphotos.baidu.com/baike/s%3D13/sign=2650abd8b2fd5266a32b3817aa184ecc/21a4462309f790523fb9e00d08f3d7ca7bcbd55b.jpg *也为偶数*

*所以*http://h.hiphotos.baidu.com/baike/s%3D13/sign=2650abd8b2fd5266a32b3817aa184ecc/21a4462309f790523fb9e00d08f3d7ca7bcbd55b.jpg*和*http://c.hiphotos.baidu.com/baike/s%3D9/sign=2c4983b00b2442a7aa0ef195d1196f/730e0cf3d7ca7bcbb85e7e1cba096b63f624a8d8.jpg*有公因数2，与*  http://d.hiphotos.baidu.com/baike/s%3D31/sign=3e9540292c34349b70066884c8eaf6e9/838ba61ea8d3fd1f5a8c0309344e251f95ca5f7f.jpg *为最简分数矛盾*

*所以*  http://a.hiphotos.baidu.com/baike/s%3D21/sign=0d0748c755da81cb4ae684cc52667118/b3fb43166d224f4a80cdd3200df790529822d1a9.jpg *为无理数*

*什么样的问题适合什么样的方法？*

*什么是理解，对自己而言就是明白是什么，对于别人而言，就是你要告诉别人这是什么。*

**跳转到Main之前，完成的初始化工作**

1. 内存空间的分配
2. 栈空间分配，SP指针初始化
3. 内存空间初始化
4. 跳转到main
5. CAN通信速率过低会导致无ack，原因？

经验之谈（操蛋）

1. 浮点数以及运算
2. ADT7410 温度监测范围
3. 以ASCII方式打开文件，将回车换行(0x0d、0x0a)（windows 64）处理成0x0a，哪里处理的？
4. 窗口看门狗
5. FPGA mcs文件格式解析
6. 地址、数据锁存？
7. 如何查看芯片自给时钟频率？

2016-8-8 16:18:37 【逻辑推理方法】

主要有三种逻辑推理的方法（更正：实际是两大类“合情推理”和“演绎推理”。合情推理又分归纳推理和类比推理）：

1、**归纳推理**　归纳是从个别对象推知一类对象，从个别性知识推知中概括出一般原理或规律的的推理形式和思维方法，归纳推理包括完全归纳法和不完全归纳法。例如在具有细胞结构的生物中，对它们的遗传物质进行推理发现，所有具有细胞结构的生物的遗传物质都是DNA，这就是完全归纳的结论。但如果把病毒也作为生物，进行遗传物质的推理发现，只有一部分病毒的遗传物质是DNA，还有一部分病毒的遗传物质是RNA，所以我们说，绝大多数生物的遗传物质是DNA，这就是一个不完全归纳的结论。细胞里面水的含量是最多的，这也是一个不完全归纳的结论，因为有极少数细胞中不的含量是很少或几乎没有水，例如小麦胚细胞中淀粉最多，脂肪细胞中的脂肪最多。

　　2、**演绎推理**　演绎是从一般到特殊，根据一类事物都有的一般属性、关系、本质来推断这类事物中的个别事物所具有的属性、关系和本质的推理形式和思维方法。

在演绎推理中，除了由一个前提推出一个结论的直接推理外，还有由两个或两个以上的前提推出一个结论的间接推理。后者中运用得比较多的是“三段论”。例如问，原子核运动不是不运动？要获得答案，可以用三段论推理：

　　大前提：物质都是运动的。

　　小前提：原子核是物质。

　　结论：原子核也是运动的。

　　值得注意的是，不完全归纳推理的结论，不能作为演绎推理的大前提。

3、**类比推理**　类比推理是逻辑推理的方法之一，它是启发人们进行创新思维的重要形式。类比推理是根据两个或两类事物在某些属性上有相同或相似之处，而且已知其中一个事物具有某种属性，由此推知另一个事物也可能具有这种属性的推理。例如，斯莱登和施旺发现植物和动物都是由细胞组成的，后来斯莱登发现了植物细胞中有细胞核，他通过类比推理，认为动物细胞中可能也有细胞核。他把这一想法告诉了施旺，后来施旺果然在动物中发现了细胞核。在科学研究中，类比推理是提出假说的重要途径，往往可以导致新发现、新理论。应当注意的是，类比推理得出的结论不一定具有逻辑上的必然性，其是否正确，还需要用其他方法来检验。

2016-8-9 13:56:52 【单片机工作原理】SP和栈

SP是CPU的一个特殊寄存器，栈是一种数据结构。

1. 目前了解到SP寄存器通常用在现场保护/子程序调用，参数传递中。SP的类型一般有两种：向上生长和向下生长，两者的区别在于压入一个元素后，SP的值是加（向上）还是减（向下）。SP里面保存的值是地址，该地址指向一片内存区域，这片内存区域的操作特点是“Last in/Fisrt out”。

此外，在早期的微机环境中，局部变量的操作是用SP来实现，但后期貌似各个平台增加了一个叫：FP寄存器（又帧寄存器）来实现局部变量的操作。

以8051为例，SP寄存器的一般操作如下

*现场保护*：如中断，在进入中断子程序前，调用PUSH指令，将CPU的PC、数据寄存器等的寄存器值依次压入SP指向的内存区域（SP+），中断处理完成后，调用POP指令依次将栈中的数据弹到对应的寄存器中。

*子程序调用*：同现场保护类似。

*参数传递*：C语言环境中，当函数参数个数超过CPU数据寄存器个数时，就需要用到SP来传递参数，具体：将传递的参数依次压入栈中，在函数中使用时，按照压入顺序依次读取。

1. SP寄存器中保存的地址

寄存器：是集成电路中最重要的一种存储单元。可以保存地址、数据和指令。

SP值（地址）：更准确的讲是栈顶地址。

栈：内存的一块区域，这块区域的操作特点是“后进先出”。（与数据结构的栈的区别和联系是？）