Preprocessor

* Thay thế đoạn define vào chỗ được gọi
* Xóa bỏ comment

Compiler

* Chyển file main.i 🡪 main.s( assembily)
* Chuyển gôn ngữ bậc cao sang ngôn ngữ bậc thấp gần giống với tập lệnh bộ vi xử lý

Asember

* Chuyển file main.s🡪 main.o(main.obj)

Linker

* 1 chương trình của mình gồm nhiều file khác nhau sẽ được sắp xếp kết hợp lại ở gian đoạn này để tạo thành 1 file cuối duy nhất để chương trình thực thi được
* Chuyển file main.o🡪 main.exe

Extern

* Kết nối các biến hoặc hàm từ các file khác nhau

Biến register

* Khi khai báo biến này nó sẽ không lưu ở stack mà lưu ở thanh ghi để tính toán

🡪tốc độ xử lý nhanh hơn,ứng dụng trong thiết kế chip…

Volatine

* Thông báo cho trình biên dịch hay optimize biết để không tối ưu code
* Sử dụng cho những biến thay đổi đột ngột mà mình kh biết trước

Static (local)

* Những biến được khai báo với từ khóa static thì nó sẽ tồn tại ngay cả khi kết thúc hàm
* Khởi tạo 1 lần duy nhất, giá trị kh mất đi khi kết thúc hàm
* Sau khi hàm được gọi, giá trị của nó chính bằng lần gần nhất hàm được gọi.
* Chỉ được sử dụng ở trong hàm khởi tạo nó.

Static (global)

* Có những tính chất như biến static cụ bộ
* Chỉ được truy cập ở file khai báo nó, các file khác không truy cập được.

Goto

* Khi chương trình phức tạp mà phải break nhiều lần mới thoát ra chương trình chính dc thì ta sẽ dùng goto

Try catch

* Cũng gần giống như If else dùng để thể hiện lỗi không mong muốn
* Try : ct kh có lỗi thì nó sẽ thực hiện
* Catch: ct báo lỗi thì nó sẽ nhảy tới catch thực hiện

Enum: nó sẽ đánh số các biến trong nhóm struct.

Ex: typedef enum{

A 0

B 1

C 2…

}

Assert: giống như 1 công cụ debug

* EX: ở những hàm hay câu lệnh quan trọng cần kiểm tra đã khai báo hay chưa thì ta dùng assert. Nếu khai báo rồi thì sẽ thông báo bug ngay chỗ đó.

Exit

* Nằm trong thư viện <stdlib.h>
* Kết thúc chương trình
* Exit(0): thoát theo dự kiến của mình
* Exit(1): chương trình bị lỗi nên thoát

Hàm atexit:

* Khi kết thúc ct nó sẽ chạy hàm này rồi nó mới kết thúc.

Ex: khi mất điện vdk sẽ thoát khỏi hàm chính và thực hiện hàm atexit rồi lưu giá trị vao bộ nhớ flash(rom).

Strcpy: copy 1 chuỗi vào trong mảng

Ex:char str[20]

Red=strcpy(str,”1234567”);

Strtod: cắt số tự nhiên của 1 chuỗi ra và trả về double, char..

Ex: char str[20]

Char resul;

Char \*\*ptr

Red=strcpy(str,”123tien”);

Resul=strtod(str,&ptr)

Strtof: char array[]={123.34 65645};

Char \*\*pend

🡪f1=strtof(array,&pend);

🡪f2=strtof(&pend,NULL);

Strtok: cắt chuỗi tại dấu hiệu cho trước

Char array[]=”luong cong tien”;

Char\*token=strtof(array,” “);

Token=strtof(NULL,” “);

* luong

cong

tien

itoa: itoa(i,array,16);

i: số muốn covers

array:mảng để lưu

16: kiểu số muốn chuyển đổi( nhị phân, thập phân,thập lục phân…)

Vector : học về bản chất của cấp phát động

C++: #include<vector>

Vector<int> mang = {1,2,3,4};

Mang.push\_bach(5);

Printf(“%d”,mang[4]);///in ra phần tử tiếp theo của mảng.

CTDL stack: thêm xóa các mảng dữ liệu trong stack

Queue: xây dựng hàng đợi

*front*: Chỉ số của phần tử ở đang đầu queue. Nó sẽ là chỉ số của phần tử sẽ bị xóa ở lần tiếp theo

*xóa*: kiểm tra rỗng hay chưa

mang[0]=’\0’;

*front++*;

*rear*: Chỉ số của phần tử tiếp theo sẽ được thêm vào cuối của queue

*thêm*: kiểm tra đầy hay chưa

mang[rear]= data;

rear++;

Thread: bản chất đa luồng

Chia nhỏ hàm trong chương trình ra thành nhiều phần. Khi chạy CT nó sẽ chạy 1 ít của hàm này và nhảy tới hàm khác chạy tiếp xen kẽ nhau như v cho đến khi chạy xong.