```
#ifndef _Y86_SIM_
#define _Y86_SIM_

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#define MAX_STEP 10000

#define BLK_SIZE 32
#define MEM_SIZE (1<<13) //MEN_SIZE=2^13;
#define REG_SIZE 32

typedef unsigned char byte_t;
typedef int long_t;
typedef unsigned char cc_t;
typedef enum { FALSE, TRUE } bool_t;

/* Y86 Condition Code */
#define GET_ZF(cc) (((cc) >> 2)&0x1)
#define GET_SF(cc) (((cc) >> 1)&0x1)
#define GET_OF(cc) (((cc) >> 0)&0x1)

#define PACK_CC(z,s,o) (((z)<<2)|((s)<<1)|((o)<<0))

#define DEFAULT_CC PACK_CC(1,0,0)


/* Y86 Register (REG_NONE is a special one to indicate no register) */
typedef enum { REG_ERR=-1, REG_EAX, REG_ECX, REG_EDX, REG_EBX,
    REG_ESP, REG_EBP, REG_ESI, REG_EDI, REG_CNT, REG_NONE=0xF } regid_t;

#define NORM_REG(_id) ((_id) >= REG_EAX && (_id) <= REG_EDI)
#define NONE_REG(_id) ((_id) == REG_NONE)

typedef struct reg {
    char *name;
    regid_t id;
} reg_t;

/* Y86 Instruction */
typedef enum { I_HALT = 0, I_NOP, I_RRMOVL, I_IRMOVL, I_RMMOVL, I_MRMOVL,
    I_ALU, I_JMP, I_CALL, I_RET, I_PUSHL, I_POPL, I_DIRECTIVE } itype_t;
```

```
/* Function code (default) */
typedef enum { F_NONE } func_t;

/* ALU code */
typedef enum { A_ADD, A_SUB, A_AND, A_XOR, A_NONE } alu_t;

/* Condition code */
typedef enum { C_YES, C_LE, C_L, C_E, C_NE, C_GE, C_G } cond_t;

/* Directive code */
typedef enum { D_DATA, D_POS, D_ALIGN } dtv_t;

/* Pack itype and func/alu/cond/dtv into single byte */
#define HPACK(hi,lo) ((((hi)&0xF)<<4)|((lo)&0xF))
#define HIGH(pack) ((pack)>>4&0xF)
#define LOW(pack) ((pack)&0xF)

#define GET_ICODE(byte0) HIGH(byte0)
#define GET_FUN(byte0) LOW(byte0)
#define GET_REGA(byte0) HIGH(byte0)
#define GET_REGB(byte0) LOW(byte0)


typedef struct mem {
	int len;
	byte_t *data;
} mem_t;

typedef struct y86sim {
	long_t pc;
	mem_t *r;
	mem_t *m;
	cc_t cc;
} y86sim_t;

#endif
```