```
#ifndef _Y86_ASM_
#define _Y86_ASM_

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

#define MAX_INSLEN    512

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


/* 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;

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

#define SIZEOF_REG    4


/* Y86 Instruction */
typedef enum { I_HALT, 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)

/* Table used to encode information about instructions */
typedef struct instr {
	char *name;
	int len;
	byte_t code; /* code for instruction+op */
	int bytes; /* the size of instr */
} instr_t;


/* Token types: comment, instruction, error */
typedef enum{ TYPE_COMM, TYPE_INS, TYPE_ERR } type_t;

typedef struct bin {
	int addr;
	byte_t codes[6];
	int bytes;
} bin_t;

typedef struct line {
	type_t type; /* TYPE_COMM: no y86bin, TYPE_INS: both y86bin and y86asm
*/
	bin_t y86bin;
	char *y86asm;

	struct line *next;
} line_t;

/* label defined in y86 assembly code, e.g. Loop */
typedef struct symbol {
	char *name;
	int addr;
	struct symbol *next;
} symbol_t;

/* binary code need to be relocated */
typedef struct reloc {
	bin_t *y86bin;
	char *name;
```

```
        struct reloc *next;
        int entry;
} reloc_t;

#endif
```