Szoftver fejlesztés

Egy adott mikroprocesszoros rendszer számára a szükséges szoftver kifejlesztése több lépésből áll:

1. Forrás nyelven megírt program(ok) lefordítása gépi kódra, amihez megfelelő fejlesztő eszközök (fordító, környezet) szükségesek.
2. Az előállított kód elhelyezése a gép memóriájában.
3. Nyomkövetés, hiba keresés.

Fordító programok

A processzor egy új fejlesztés, így nincsenek hozzá kész, felhasználható fordítók. A programok lefordításához készült egy egyszerű assembler program, amivel assembly nyelven megírt programot le lehet fordítani a memóriában való elhelyezéshez szükséges formára.

Assembler

Az assembler egy online web-es felületen keresztül érhető el:

http://mazsola.iit.uni-miskolc.hu/mem/asm

Az oldalt megnyitva egy beviteli területet kapunk, ahová be kell gépelnünk a programunk assembly (forrás) kódját. A mező alatti `Download` gombra kattintva megkapjuk a lefordított fájlt, ugyanezt a `View` gombra kattintva letöltés nélkül nézhetjük meg.

Az assembly forrás szintaxisa

A forráskódot, más assembly nyelvekhez hasonlóan, soronként, egy utasítást egy sorba írva kell összeállítanunk. Egy sor szerkezete a következő lehet:

```
[cimke:] [feltétel] utasítás paraméterek [; megjegyzés]
```

A `cimke:` (a szó végi kettőspont jelöli) egy névvel látja el azt a memória helyet, ahová az utasítás kerül. Ugrásoknál és szubrutin hívásoknál használható a cím megadására.

Az utasítások feltétellel is elláthatók, ha ezt nem adjuk meg, akkor az utasítás feltétel nélkül hajtódik végre. A feltételek a következők lehetnek: `C0, C1, O0, O1, Z0, Z1, S0, S1`, tehát a jelzőbit nevéből, és a szükséges értékéből állhatnak. A `Z0` feltételt írhatjuk `NZ`, a `Z1`-et pedig `Z` alakban is.

Az utasítás neve a mikroprocesszor leírásában szereplő utasítás név lehet.

A paraméterek az utasítástól függenek, ha több paramétert kell megadnunk, akkor azokat vesszővel választjuk el egymástól. A szükséges paraméterekről szintén az utasítások

leírásából tájékozódhatunk. Regiszterek megadásánál a `R` betűt és a regiszter sorszámát használjuk. Adatok esetében használhatunk címkeként létrehozott szimbólumokat, vagy konstansokat. A konstansoknak számjeggyel kell kezdődniük. A `0`-val kezdődő konstansokat a fordító 8-as számrendszerűnek tekinti, míg a `0x`-el kezdődőeket 16-osnak.

A megjegyzéseket pontosvesszővel kell kezdeni, innen a sor hátralévő része megjegyzés lesz, amit a fordító figyelmen kívül hagy.

Assembly utasítások

Bizonyos utasítások nem a processzor utasításainak a leírására szolgálnak, hanem a fordító működését szabályozzák, vagy egyéb eredményük van.

ORG érték

Az ORG utasítás paramétere egy számérték, ez lesz a memória cím számláló értéke. Arra használható, hogy valamilyen tartalmat egy adott memória területen helyezzünk el.

```
        org     0       ; 0-s címen elhelyezve
        call    rutin
vege:   jmp     vege

        org     50      ; a rutint az 50-es címre tesszük
rutin:
        ldl0    r1,0    ; ....
```

szimbólum EQU érték

Ezzel az utasítással egy szimbólumnak értéket adhatunk. Az `ORG` kulcsszó helyett az `=` jelet is használhatjuk.

```
porta   =       0xf000          ; a kimeneti port címe

        ldl     r0,porta        ; a cím betöltése
        ldh     r0,porta        ; az R0 regiszterbe
```

DS érték

Ez az utasítás hely foglalásra használható. Ha a memóriában le akarunk foglalni valahány rekeszt adat tárolásra, ezzel az utasítással megadhatjuk, hány szó helyre van szükségünk. Ezek a memória helyek 0-val töltődnek fel.

```
        org     20              ; ettől a címtő kezdve
adatok: ds      5               ; 5 szó helyet foglalunk le
```

DB,DW,DD érték

Ezekkel az utasításokkal a paraméterként megaadott értéket tudjuk elhelyezni a memóriában. A `DB` az érték alsó 8 bitjét használja fel, a `DW` 16 bitet, míg a `DD` 32 bites adatokat használ.

```
start:  ldl0    r1,0
        ...

        org     10
adat1:  db      123             ; 8 bites adat
        dw      0x213           ; 16 bites adat
        dd      -4321           ; negatív számok is használhatók
        dd      start           ; vagy akár címkék is
```

Kényelmi szolgáltatások

Paraméterek elhagyása

Gyakori eset, hogy két operandusú aritmetikai műveletnél (pl `INC`, `SHL`, stb.) a két operandus ugyanaz a regiszter. Ezeknél az utasításoknál elég az egyik paramétert megadni, a fordító a másikat automatikusan kiegészíti.

```
        inc     r0,r1           ; r0:= r1+1
        inc     r2,r2           ; r2:= r2+1
        inc     r2              ; így is írható
```

Regiszter nevek

A `R15` regiszter helyett használhatjuk a `PC`, az `R14` helyett az `LR`, az `R13` helyett pedig az `SP` elnevezést is.

```
start:  ldl0    pc,ide          ; betöltés az R15-be (ugrás)
        ...

ide:    ldl     sp,stack        ; R13 beállítása
        ldh     sp,stack

        st      lr,sp           ; az R14 mentése
        ...

stack:  org     100             ; itt lesz a stack helye
```

Utasítás nevek

Egyes utasítások más nevekkel is megadhatók, esetleg rövidített paraméterezéssel:

Alternatív név	Lefordított utasítás	Funkció
`JMP adat`	`LDL0 R15,adat`	Ugrás az alsó 64 Ki-ban
`JZ adat`	`Z1 LDL0 R15,adat`	Feltételes ugrás (ha zero)
`JNZ adat`	`Z0 LDL0 R15,adat`	Feltételes ugrás (ha nem zero)
`JP reg`	`MOV R15,reg`	Indirekt ugrás
`RET`	`MOV R15,R14`	Ugrás az LR-be mentett címre (visszatérés szubrutinból)
`PUSH reg`	`ST reg,R13`	Regiszter írása az R13-ban lévő címre
`POP reg`	`LD reg,R13`	Regiszter beolvasása az R13-ban lévő címről

A kód beépítése a rendszerbe

Az FPGA-n kialakított teszt áramkör nem tartalmaz külső hozzáférési lehetőséget a memóriához, vagyis a JTAG, vagy hasonló módszerekkel való kód letöltés nem használható. Mivel a memória nem külső áramkörként kapcsolódik az FPGA csiphez, hanem azon belül van kialakítva, ezért a program kódját bele kell építeni az FPGA terv (projekt) memória alkatrészébe, majd újra kell fordítani az FPGA tervet.
Az FPGA projektet a Xilinx cég ISE tervező rendszerével készítettük. Ebben a memória alkatrészt Verilog nyelvű modulként illesztettük be. Ebben a kezdőérték beállító paraméterekben elhelyezhető a szükséges programkód.
Ennek a rendszernek a támogatására a online assembler fordító a lefordított kódot közvetlenül az FPGA proketbe illeszthető Verilog modul formájában állítja elő.

A szükséges lépések

1. Nyissuk meg a http://mazsola.iit.uni-miskolc.hu/mem/asm címen lévő oldalt.
2. Írjuk be a programunk assembly nyelvű forrását.
3. Nyomjuk meg a **`Download verilog file`** gombot.
4. A letöltött fájlt mentsük el `mem.v` néven közvetlenül az ISE projekt könyvtárába (felülírva az esetleg ott lévő régebbi változatot).
5. Fordítsuk le az ISE projektet.
6. Az **adept** program segítségével töltsük fel a lefordított `bit` fájlt a kártyára.

Nyomkövetés

A program kóddal kiegészített és lefordított teszt áramkört a kártyán lévő FPGA-ba betöltve a program működését tesztelhetjük. A teszt áramkör kiinduló helyzetben a `BTN0` nyomógombot kapcsolja a processzor órajel bemenetére, a kártya kijelzőin pedig a diagnosztikai jeleket jeleníti meg. Ezeknek a részletes ismertetését lásd a teszt áramkör leírásánál.
Ezt a rendszert felhasználhatjuk a program részletes nyomkövetésére, nagyobb frekvenciájú órajel kiválasztásával pedig a futtatására is.