Döntő

2009. február 28.

Processzor

Az Intal cég új processzort kíván gyártani. A processzor fantázianeve: Intal Pentimum.

A tervek elkészültek, de mielőtt a sorozatgyártásba kezdenének, meg akarnak győződni arról, hogy a Pentimum tervezett utasításkészlete megfelel a felhasználói igényeknek.

A Pentimumot egy olyan alaplapra integrálták, amelynek egy IO-felülete a hozzá csatlakoztatott $400[1] byte kapacitású memória, egy outputja pedig egy képernyő illesztési felület. A képernyő felbontása 10x10 karakter, a képernyő csak szöveges információ megjelenítésére alkalmas.

A csapat feladata, hogy szoftveres támogatást nyújtson a processzor funkcionalitásának és architektúrájának[2] a kipróbálásához, teszteléséhez.

Feladatok:

- **fordítóprogram készítése**:
  A program bemenete egy szövegfile, amely a Pentimum utasításkészletén implementált[3] programot ír le assembly[4] nyelven.
  A fordítóprogram ezt az assembly programot fordítja le gépi kódra a processzor megadott utasításkészletének megfelelően.
  A program kimenete a gépi kódot tartalmazó szövegfájl.

- **emulátor készítése**:
  Ennek a programnak a segítségével szoftveresen emulálható[5] a processzor működése. Az emulátor bemenete egy byte-kódú[6] (gépi kódú) program. Az emulátor PC-n futtatható, és a PC képernyőjén jeleníti meg a memória és a processzor állapotát.

- **mintaprogram készítése**:
  A cég mintaprogramot is kíván adni a processzor dokumentációja mellé, ezért egy kitűzött feladatot is meg kell oldani a processzor utasításkészletével.
  A mintaprogram egy olyan assembly kód, amelyet a processzor utasításkészletével kell megírni, a fordítóprogrammal le kell fordítani, majd az így kapott gépi kódú programot kell az emulátoron futtatni.

A Pentimum és architektúrája

A memória mérete $400 byte. Az első $64 byte a 10x10 karaktert megjelenítő képernyő számára fenntartott memória. Minden egyes byte egy cellát jelöl a képernyőn, az ott lévő értékek pedig – a

---

[1] A $400 hexadecimális adat (decimálisan: 1024)
[2] **Architektúra:** terv, koncepció
[3] **Implementál:** megvalósít
[4] Az **assembly nyelv** a gépi kód szimbolikus megfelelője, azért jött létre, hogy könnyebben lehessen programozni gépi kód szinten. Gépközeli nyelvnek is szokták nevezni.
[5] **Emulál**: utánoz (Az emuláció lényege, hogy az emulált környezethez készült szoftverek és adatok feldolgozását lehetővé teszi az attól eltérő jellemzőkkel rendelkező környezetben is.)
[6] **Byte-kód**: Utasításssorozat, amelyben minden utasítás egy vagy több egymást követő byte-ban található. Az egyes byte-okban szereplő adatokat leggyakrabban kettes vagy tizenhatos számrendszerben írják le.

megjelenítés szempontjából – ASCII kódolásúak. A memóriahelyek és a 10x10-es képernyő kapcsolata:

$00	$01	$02	$03	$04	$05	$06	$07	$08	$09
$A	$B	$C	$D						$13
$14	$15	$16							$1D
$1E	$1F								$27
$28									$31
$32									$3B
$3C									$45
$46									$4F
$50									$59
$5A	$5B	$5C	$5D	$5E	$5F	$60	$61	$62	$63

A memória a $64. byte-tól kezdve az $3FF. byte-ig tetszőlegesen használható. A képernyő mind a $64 byte-ja kezdetben a $20-as byte-tal (szóköz) van feltöltve.

Az emulátorba töltött program byte-jai a memória $64. byte-jától kerülnek be, és a program futtatása is onnan kezdődik.

A Pentimum utasításkészlete nagyon egyszerű. A processzornak nincsenek regiszterei, a műveleteket mindig 8 bites byte-okon végzi.

Az utasításkészlet

A címzés hexadecimálisan történik.

Assembly utasítás	**Leírás**	**Byte-kód**
MOV [ABC] [DEF]	A DEF memóriahelyen lévő 8 bites érték átmásolása az ABC memóriahelyre.	**00 0A BC 0D EF** (5 byte)
pl. MOV [A1] [113]	A $113-as (decimálisan 275-ös) memóriahelyen lévő byte az A1-es (decimálisan 161-es) címre kerül.	00 00 A1 01 13
MOV [ABC] $XY	Az $XY byte bemásolása az ABC memóriahelyre.	**01 0A BC XY** (4 byte)
pl. MOV [A] $48	A 'H' karakter hexakódja az A (decimálisan 10-es) memóriacímre kerül.	01 00 0A 48

ADD [ABC] [DEF]	Az ABC és DEF memóriahelyen lévő két 8 bites szám összeadása. A 8 bites eredmény az ABC memóriahelyre kerül.	**02 0A BC 0D EF** (5 byte)
ADD [ABC] $XY	Az ABC memóriahelyen lévő 8 bites érték ill. az $XY érték összeadása. A 8 bites eredmény az ABC memóriahelyre kerül.	**03 0A BC XY** (4 byte)
SUB [ABC] [DEF]	Az ABC memóriahelyen lévő 8 bites számból a DEF memóriahelyen 8 bites szám kivonása. A 8 bites eredmény az ABC memóriahelyre kerül.	**04 0A BC 0D EF** (5 byte)
SUB [ABC] $XY	Az ABC memóriahelyen lévő 8 bites értékből az $XY érték kivonása. A 8 bites eredmény az ABC memóriahelyre kerül.	**05 0A BC XY** (4 byte)
JMP ABC	A következő utasítás végrehajtása az ABC címtől folytatódik.	**06 0A BC** (3 byte)
JZ ABC	A következő utasítás végrehajtása az ABC címtől folytatódik, ha az utolsó MOV, ADD vagy SUB utasítás eredménye 0 volt, egyébként tovább folytatódik a végrehajtás.	**07 0A BC** (3 byte)
JC ABC	A következő utasítás végrehajtása az ABC címtől folytatódik, ha az utolsó ADD vagy SUB utasítás eredmény e túl- ill. alulcsordult, egyébként tovább folytatódik a végrehajtás.	**08 0A BC** (3 byte)
WAIT	A processzor 100ms-ig várakozik	**09** (1 byte)
END	A processzor megáll. (A program véget ér.)	**0A** (1 byte)

A WAIT utasításon kívüli összes utasítás végrehajtási ideje elhanyagolható (0ms) a WAIT utasítás 100ms-os végrehajtási idejéhez képest.

Az assembly utasítások paraméterei között egy-egy szóköz van. Az utasítások azonosítására szolgáló paraméter minden betűje nagybetű.

Az utasításkészlet táblázatában a byte-kódok egyes byte-jait szóköz választja el. Az emulátor program bemeneteként egy olyan szövegfájlt kell készíteni, amelyben az egyes utasítások összetartozó byte-jai, és az egymást követő utasítások szomszédos byte-jai elválasztójel nélkül követik egymást.

A fordítóprogram

A fordítóprogram bemeneti állománya (egy assembly-forráskódú program) soronként egy-egy utasítást tartalmaz a fenti Pentimum utasításkészlet alapján. A forráskódban lehetőség nyílik kommentezésre: amennyiben egy sor a # karakterrel kezdődik, azt a fordítóprogram figyelmen kívül hagyja.

A fordítóprogram által készített gépi-kódú program byte-kódja szöveges állományba kerül kiírásra úgy, hogy az egymást követő byte-ok között **semmilyen elválasztójel** nincsen. Minden egyes byte-ot hexadecimálisan kell kiírni.

(Javaslat: A fordítóprogram az alapértelmezett kimenet mellett hozzon létre egy másik kimenő fájlt is. Ebben az egyes utasításokhoz tartozó byte-kódok kerüljenek külön sorba, és az egyes byte-ok a soron belül egy-egy szóközzel legyenek elválasztva. Az utasítások byte-kódjainak elválasztása a fordítóprogram ellenőrzését teszi könnyebbé.)

Ha a fordítóprogram fordítás közben hibát talál, meg kell állnia, és hibaüzenetet kell megjelenítenie.

A lehetséges hibák:

- szintaktikai hiba az assembly programban (az assembly utasítás nem a megadott szerkezetű),
- túl hosszú a program (nem fér be a memóriába).
- ha olyan memóriacímre történik hivatkozás, ami nem létezik.

A hibára vonatkozó figyelmeztetést és a hibás sor sorszámát a képernyőre kell kiíratni. Pl. „Szintaktikai hiba a 15. sorban.”, vagy „A program nem fér be a memóriába. Hiba a 40. sorban.” (Itt az assembly program sorának száma decimális adatként jelenjen meg!)

A fordítóprogram leállása esetén a kimeneti állomány tartalmának nincs jelentősége. (Nem szükséges azt elkészíteni.)

Az emulátor működése

Az emulátor az elindításakor kér egy byte-kódot (szöveges bemeneti állományt), ennek az adataival azonnal fel is tölti a Pentimum memóriájának a gépi kódú program számára rendelkezésre álló részét.

Az emulátor működése ezután az alábbi parancsokkal vezérelhető:

- **„L fájlnév” (Load):** az emulátor betölti a megadott filenévhez tartozó file-ban lévő programot, és inicializálja magát;

- **„R" (Run):** az emulátor megállás nélkül folytatja a program futtatását;
    - a futtatás tetszőleges billentyű lenyomásával félbeszakítható, az emulátor a továbbiakban az említett utasításokkal vezérelhető;
- **„S" (Step):** az emulátor végrehajtja a soron következő utasítást;
- **„M 123" (Memory):** a megadott memórián lévő értéket megjeleníti. A memóriacímet hexadecimálisan kell megadni.
- **„Q" (Quit):** kilépés az emulátorból.

Az emulátor futtatás közben három ok miatt állhat le:

- END utasításhoz ér,
- lefagy
    - ha az emulátor a memória végéhez ér,
    - ha az emulátor olyan utasítást kísérel meg végrehajtani, ami nincs az utasításkészletben,
- a futtatását felhasználói közreműködéssel félbeszakítják.

Az R (Run) és S (Step) utasítások csak akkor hajthatók végre, ha a processzor nem fagyott le, vagy az emulátor program nem ért el egy END utasítást. Leállás után az emulátor csak az L (Load) utasítással hozható alaphelyzetbe.

Az emulátornak külön kérés nélkül mutatnia kell a következő információkat:

- az aktuális 10x10-es képernyő tartalmát,
- a végrehajtás alatt álló vagy az éppen végrehajtott utasítás byte-kódját és memóriacímét.

Feltételezhetjük, hogy a bemeneti byte-kód nem tartalmaz szintaktikai hibát.

Példa

Assembly forráskód:
példa (Hello World):

```
# Print „HELLO"
MOV [0] $48
MOV [1] $45
MOV [2] $4C
MOV [3] $4C
MOV [4] $4F
# Print „WORLD"
MOV [A] $57
MOV [B] $4F
MOV [C] $52
MOV [D] $4C
MOV [E] $44
END
# End of program
```

A lefordított szöveges állományú gépi byte-kód:
A teszteléshez:

```
01 00 00 48
01 00 01 45
01 00 02 4C
01 00 03 4C
01 00 04 4F
01 00 0A 57
01 00 0B 4F
01 00 0C 52
01 00 0D 4C
01 00 0E 44
0A
```

Az emulátor bemenetként megkapja az egy sorban megjelenített byte-kódot:

0100004801000145010002 4C0100034C0100044F01000A5701000B4F01000C5201000D4C01000E440A

A felhasználóbarát felületen az emulátor vezérlési lehetőségeinek rövid leírása is megjelenik.

Az emulátor felhasználói felületét tekintve nem elvárás a grafikus megjelenítés. Az emulátor programban a karakteres és a grafikus megjelenítés azonos értékűnek számít.

A mintaprogram

A processzor mellé adott assembly-mintaprogramnak a következőt kell megvalósítania.

A képernyő 0. sorában a „Dusza” feliratot kell képújságszerűen görgetni. A képernyő 0. sora először üres, majd minden eltelt 500ms után megjelenik egy újabb karakter úgy, hogy az eddig megjelent szöveg egy karakternyit balra mozdul. A következő táblázat mutatja a 0. sor tartalmát az idő függvényében:

Idő/oszlop	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.
0ms										
500ms										D
1000ms									D	u
1500ms								D	u	s
2000ms							D	u	s	z
2500ms						D	u	s	z	a
3000ms					D	u	s	z	a	
3500ms				D	u	s	z	a		
4000ms			D	u	s	z	a			
4500ms		D	u	s	z	a				
5000ms	D	u	s	z	a					
5500ms	u	s	z	a						
6000ms	s	z	a							
6500ms	z	a								
7000ms	a									
7500ms										

A 7500. ms egyben a képújság újraindulása is: a 8000. ms-ban ugyanaz folytatódik, mint ami az 500. ms-ban elkezdődött.

A képújság görgetése közben a képernyő (a Pentimum képernyőjének) bal alsó sarkában jelenjen meg egy másodperc számláló óra is, amely a program elindítása után eltelt másodperceket számolja. (Ez az óra nem az emulátornak, hanem az elkészítendő assembly-programnak a része.)

Az elkészített assembly-forráskódnak fordíthatónak kell lennie a fordítóval, majd az elkészült byte-kódot az emulátornak képesnek kell lennie futtatni úgy, hogy a PC-képernyőjén nyomon követhető legyen a megírt assembly-program működése (látszódjon a képújság és az óra is).

Beadandó:

- A fordítóprogram forráskódja és a lefordított állomány
- Az emulátor forráskódja és a lefordított állomány
- A fejlesztői dokumentáció (A megadott sablon alapján elektronikusan kell elkészíteni.)
- A mintaprogram assembly kódja (Elektronikusan kell elkészíteni.)

A bemutatáshoz külön szemléltető anyag készítése (pl. PowerPoint prezentáció) nem kötelező, de ha készül ilyen, azt is be kell adni!

A munka szóbeli bemutatása:

Szempontok, ajánlott vázlat:

- A feladat előkészítésének bemutatása, a feladatok szétosztásának elvei
- Az elkészített fordítóprogram és emulátor bemutatása
  - a felhasználó számára

    A programok használatát kell bemutatni a megadott tesztfájlok használatával.

    (A programok írásakor a teszteléshez használt assembly kódok és byte-kódok mellett a zsűri által a bemutatón rendelkezésre bocsátott további tesztekkel is le kell futtatni a fordítót és az emulátort.)
  - a fejlesztő számára

    A programok szerkezetét kell bemutatni.
- a mintaprogram bemutatása

Minden csapattagnak részt kell vennie a munka bemutatásában!

Elérhető pontszám: 140 pont