Fordítóprogramok beugrókérdések

1) **Mirol mire fordít a fordítóprogram?**
Általában magas szintű programozási nyelvről gépi kódra.

2) **Mirol mire fordít az assembler?**
Assembly nyelvről gépi kódra.

3) **Mi a különbség a fordítóprogram és az interpreter között?**
Fordítóprogram: A forráskódot a fordítóprogram tárgyprogrammá alakítja. A tárgyprogramokból a szerkesztés során futtatható állomány jön létre.
Interpreter: A forráskódot az interpreter értelmezi és azonnal végrehajtja.

4) **Mi a virtuális gép?**
Olyan gép szoftveres megvalósítása, amelynek a bájtkód a gépi kódja.

5) **Mi a különbség a fordítási és a futási ido között?**
Fordítási idő: amikor a fordító dolgozik
Futási idő: amikor a program fut

6) **Mi a feladata az analízisnek (a fordítási folyamat elso fele) és milyen részfeladatokra bontható?**
Az analízis elsődleges feladata, hogy meghatározza az egyes szimbolikus egységeket, konstansokat, változókat, kulcsszavakat, operátorokat ezt a feladatot a lexikális elemző végzi, ami egy karaktersorozatból szimbólum sorozatot generál.
Ez bekerül a szintaktikus elemzőbe, aminek a feladata a program struktúrájának felismerése, ellenőrzése: szimbólumok megfelelő sorrendben vannak-e? értelmes-e? a nyelvnek megfelel-e? stb. ezt megkapja a szemantikus elemző, mely eldönti, a kód a szükséges szemantikus tulajonságokkal rendelkezik-e?

7) **Mi a feladata a szintézisnek (a fordítási folyamat második fele) és milyen részfeladatokra bontható?**
Szintézis feladatainak részletezése:
az első lépés a kódgenerálás, melyet a kódgenerátor végez ezt követi a kódoptimalizálás.

8) **A fordítóprogram mely részei adhatnak hibajelzést?**
A lexikális, szintaktikus, szemantikus elemzők adhatnak hibát.

9) **Mi a lexikális elemző feladata, bemenete, kimenete?**
Feladat: lexikális egységek (szimbólumok) felismerése – azonosító, konstans, kulcsszó,…
input: karaktersorozat
output: szimbólumsorozat, lexikális hibák

10) **Mi a szintaktikus elemző feladata, bemenete, kimenete?**
Feladat: program szerkezetének felismerése, a szerkezet ellenőrzése: megfelel-e a nyelv definíciójának?
input: szimbólumsorozat
output: szintaxisfa, szintaktikus hibák

11) **Mi a szemantikus elemző feladata, bemenete, kimenete?**
Feladat: típusellenőrzés, válrtozók láthatóságának ellenőrzése, eljárások hívása megfelel-e a szignatúrának?, stb.
input:szintaktikusan elemzett program
output: szemantikusan elemzett program, szemantikus hibák.

12) **Mi a kódgenerátor feladata, bemenete, kimenete?**
Feladat: forrásprogrammal ekvivalens tárgyprogram készítése
input: szemantikusan elemzett program
output: tárgykód utasításai (pl assembly, gépi kód)

13) Mi a kódoptimalizáló feladata, bemenete, kimenete?
Feladat:valamilyen szempontok szerint jobb kód készítése:
- futási sebesség növelése, méret csökkentése,
- felesleges utasítások megszüntetése, ciklusok kifejtése….

input: tárgykód
output: optimalizált tárgykód

14) Mi a fordítás menetszáma?
A fordítás annyi menetes, ahányszor a programszöveget (vagy annak belső reprezentációit) végigolvassa a fordító a teljes fordítási folyamat során.

15) **Milyen nyelvtanokkal dolgozik a lexikális elemző?**
reguláris, 3-as nyelvtanokkal

16) Hogy épülnek fel a reguláris kifejezések?
Elemek: üres halmaz, üres szöveget tartalmazó halmaz, egy karaktert tartalmazó halmaz
Konstrukciós műveletek a fenti halmazokkal: konkatenáció, unió, lezárás
További „kényelmi” műveletek: +, ?

17) Hogy épülnek fel a véges determinisztikus automaták?
Elemei: ábécé, állapotok halmazai, átmenetfüggvény, kezdőállapot, végállapotok halmaza

18) **Milyen elv szerint ismeri fel a lexikális elemző a lexikális elemeket?**
A lexikális elemző mindig a lehető leghosszabb karaktersorozatot ismeri fel.

19) Mi a szerepe a lexikális elemek sorrendjének?
A lexikális elemző megadásakor sorbarendezhetjük a szimbólumok definícióit.
Ha egyszerre több szimbólum is felismerhető, a sorrendben korábbi lesz az eredmény.

20) Mi a különbség a kulcsszavak és a standard szavak között?
A standard szó felüldefiniálható, de a kulcsszavak nem.

21) **Mi az elofeldolgozó fázis feladata?**
- feljegyzi a makródefiníciókat,
- elvégzi a makróhelyettesítéseket,
- meghívja a lexikális elemzőt a beillesztett fájlokra,
- kiértékeli a feltételeket és dönt a kódrészletek beillesztéséről vagy törléséről.

22) **Mutass példát olyan hibára, amelyet a lexikális elemző fel tud ismerni és olyanra is, amelyet nem!**
Feltud ismerni:
-hibás számformátum (pl. 1.23.45)
valamelyiket illegális karakternek lehet tekinteni

Nem tud felismerni:
-kihagyott szimbólum (pl. 1+a helyett 1a, a+1 helyett a1)
ezeket csak a szintaktikus elemző tudja észrevenni

23) Mikor ciklusmentes egy nyelvtan?

Egy nyelvtan ciklusmentes, ha nincs A ⇒+ A levezetés

Ellenpélda: A->B, B->A

24) Mikor redukált egy nyelvtan?

„nincsenek felesleges nemterminálisok”

- minden nemterminális szimbólum előfordul valamelyik mondatformában
- mindegyikből levezethető valamelyik terminális sorozat
  ellenpélda: A->aA, ha ez az egyetlen szabály A-ra

25) Mikor egyértelmu egy nyelvtan?

Minden mondathoz pontosan egy szintaxisfa tartozik.

26) **Mi a különbség a legbal és legjobb levezetés között?**

Legbal: Mindig a legbaloldalibb nemterminálist helyettesítjük. S=>AB=>aaB=>aab
Legjobb: Midig a legjobboldalibb nemterminálist helyettesítjük. S=>AB=>Ab=>aab

27) Mi a különbség a felülrol lefelé és az alulról felfelé elemzés között?

Felülről lefelé:
A startszimbólumból indulva, felülről lefelé építjük a szintaxisfát. A mondatforma baloldalán megjelenő terminálisokat illesztjük az elemzendő szövegre.

Alulról felfelé:
Az elemzendő szöveg összetartozó részeit helyettesítjük nemterminális szimbólumokkal (redukció) és így alulról, a startszimbólum felé építjük a fát.

28) **Mi az összefüggés az elemzési irányok és a legbal, illetve legjobb levezetés között?**

Felülről lefelé elemzés = Legbal levezetés
Alulról felfelé levezetés = Legjobb levezetés inverzével.

29) Milyen alapveto stratégiák használatosak a felülrol lefelé elemzésekben?

- visszalépéses keresés, előreolvasás

30) Milyen alapveto stratégiák használatosak az alulról felfelé elemzésekben?

- visszalépéses keresés, precedenciák használata, előreolvasás

31) Definiáld a FIRSTk(_) halmazt, és röviden magyarázd meg a definíciót!

Az **α** mondatformából levezethető terminális sorozatok **k** hosszúságú kezdőszeletei.

(ha a sorozat hossza kisebb mint k, akkor az egész sorozat eleme FIRSTk (α)-nak, akár $\varepsilon \in$ FIRSTk

(α) is előfordulhat)

FIRSTk (α) = $\{x \mid \alpha \Rightarrow^* x\beta \wedge |x| = k\} \cup \{x \mid \alpha \Rightarrow^* x \wedge |x| < k\}$

32) Definiáld az LL(k) grammatikákat és röviden magyarázd meg a definíciót!

A levezetés tetszőleges pontján a szöveg következő **k** terminálisa meghatározza az alkalmazandó levezetési szabályt.

Tetszőleges

$$S \Rightarrow^* wA\beta \Rightarrow w\,\alpha 1\,\beta \Rightarrow^* wx$$

$$S \Rightarrow^* wA\beta \Rightarrow w\,\alpha 2\,\beta \Rightarrow^* wy$$

levezetéspárra $FIRSTk(x) = FIRSTk(y)$ esetén $\alpha 1 = \alpha 2$.

33) Definiáld a FOLLOWk(_) halmazt és röviden magyarázd meg a definíciót!

$$FOLLOWk(\alpha) = \{x \mid S \Rightarrow^* \beta\alpha\gamma \wedge x \in FIRSTk(\gamma) \setminus \{ \varepsilon \}\} \cup \{\# \mid S \Rightarrow^* \beta\alpha\}$$

FOLLOWk (α): a levezetésekben az α után előforduló k hosszúságú terminális sorozatok (ha a sorozat hossza kisebb mint **k**, akkor az egész sorozat eleme FOLLOWk (α)-nak,

ha **α** után vége lehet a szövegnek, akkor # ∈ FOLLOWk (α))

34) **Definiáld az egyszerű LL(1) grammatikát!**

Olyan LL(1) grammatika, amelyben a szabályok jobboldala terminális szimbólummal kezdődik (ezért ε-mentes is). (Az összes szabály A → aα alakú.)

35) **Mi az egyszerű LL(1) grammatikáknak az a tulajdonsága, amire az elemző épül?**

Egyszerű LL(1) grammatika esetén az azonos nemterminálishoz tartozó szabályok jobboldalai különböző terminálissal kezdődnek.

36) **Mit csinál az egyszerű LL(1) elemző, ha a verem tetején az A nemterminális van és a bemenet következo szimbóluma az a terminális?**

Ha van A → aα szabály:

- A helyére aα és bejegyzés a szintaxisfába
- különben: hiba

37) **Definiáld az ε -mentes LL(1) grammatikát!**

Olyan LL(1) grammatika, amely ε-mentes. (Nincs A → ε szabály.)

38) **Mi az ε -mentes LL(1) grammatikáknak az a tulajdonsága, amire az elemző épül?**

ε-mentes LL(1) grammatika esetén az egy nemterminálishoz tartozó szabályok jobboldalainak FIRST1 halmazai diszjunktak.

39) **Mit csinál az Ɛ -mentes LL(1) elemző, ha a verem tetején az A nemterminális van és a bemenet következo szimbóluma az a terminális?**

Ha van A → α szabály, amelyre a ∈ FIRST1 (α):

- A helyére α és bejegyzés a szintaxisfába
- különben: hiba

40) **Definiáld az LL(1) grammatikát!**
Tetszőleges

S ⇒* wAβ ⇒ w α1 β ⇒* wx

S ⇒∗ wAβ ⇒ w α2 β ⇒∗ wy

levezetéspárra FIRST1 (x) = FIRST1 (y) esetén α1 = α2 .

41) **Mi az LL(1) grammatikáknak az a tulajdonsága, amire az elemző épül?**
FIRST1 (αFOLLOW1 (A)) jelentése: α-hoz egyenként konkatenáljuk
FOLLOW1 (A) elemeit és az így kapott halmaz minden elemére alkalmazzuk a FIRST1 függvényt.

42) **Mit csinál az LL(1) elemző, ha a verem tetején az A nemterminális van és a bemenet következo szimbóluma az a terminális?**

Ha van A → α szabály, amelyre a ∈ FIRST1 (αFOLLOW1 (A)):

- A helyére α és bejegyzés a szintaxisfába
- különben: hiba
-

43) **Milyen komponensei vannak az LL(1) elemzőknek?**
//táblázat miből épül fel

44) **Hogyan épülnek fel a rekurzív leszállásos elemzőben a nemterminális szimbólumokhoz rendelt eljárások?**
Nemterminális szimbólumokhoz rendelt eljárások a nyelvtani szabálytól függűen tartalmaznak más eljárásokat amik nyelvtani szabályokat reprezentálnak, vagy terminális szimbólun levezetéseket, a szabálynak megfelelő sorrendben.
Ezek elágazásokban helyezkednek el az éppen aktuálisan olvasott szimbólum szempontjából.

45) **Mit jelentenek a léptetés és redukálás műveletek?**
Redukálás: Az elemzendő szöveg összetartozó részeit helyettesítjük nemterminális szimbólumokkal (redukció) és így alulról, a kezdőszimbólum felé építjük a fát.
Léptetés: Egyik állapotból a másikba lépünk az automatával.

46) Mi a kiegészített grammatika és miért van rá szükség?
Az elemzés végét arról fogjuk felismerni, hogy egy redukció eredménye a kezdőszimbólum lett.

Ez csak akkor lehet, ha a kezdőszimbólum nem fordul elő a szabályok jobboldalán.
Ezt nem minden grammatika teljesíti, de mindegyik kiegészíthető:
legyen S' az új kezdőszimbólum
legyen $S' \rightarrow S$ egy új szabály

47) Mi a nyél szerepe az alulról felfelé elemzésekben?
Nyél: a mondatformában a legbaloldalibb egyszerű részmondat.
Épp a nyelet kell megtalálni a redukcióhoz.

48) Mondd ki az LR(k) grammatika definícióját és magyarázd meg!
Egy kiegészített grammatika LR(k) grammatika ($k \geq 0$), ha

$$S \Rightarrow^* \alpha A w \Rightarrow \alpha \beta w$$

$$S \Rightarrow^* \gamma B x \Rightarrow \gamma \delta x$$

$\alpha\beta y = \gamma\delta x$ és FIRSTk (w) = FIRSTk (y) esetén
$\alpha = \gamma$, $\beta = \delta$ és A = B.

49) **Hogyan határozza meg az LR(0) elemző véges determinisztikus automatája, hogy léptetni vagy redukálni kell?**
Az átmeneteit a verembe kerülő szimbólumok határozzák meg:
- léptetéskor terminális
- redukáláskor nemterminális

amikor elfogadó állapotba jut, akkor kell redukálni!

50) Hogy néz ki egy LR(0)-elem és mi a jelentése?
Ha $A \rightarrow \alpha$ a grammatika egy helyettesítési szabálya, akkor az $\alpha = \alpha 1\ \alpha 2$ tetszőleges felbontás esetén $[A \rightarrow \alpha 1\ .\alpha 2\ ]$ a grammatika egy LR(0) eleme.

51) **Milyen műveletek segítségével állítjuk elő a kanonikus halmazokat és mi ezeknek a szerepe?**
closure (lezárás) és read (olvasás) műveletek segítségével állítjuk elő a kanonikus halmazokat.
Ezen műveletek akalmazásával tudjuk meghatározni a kanonikus halmazokat => az adott állapotból melyik állapotba tudunk továbbmenni.

52) **Mi köze van az LR(0) kanonikus halmazoknak az LR(0) elemző véges determinisztikus automatájához?**
A végállapotok azok a kanonikus halmazok, amelyekben olyan elemek vannak, ahol a pont a szabály végén van.

53) **Hogyan határozzuk meg az LR(0) elemző automatájában az átmeneteket?**
closure read kanonikus halmazokat alakítunk ki, ha nem végállapot van a halmazban, akkor léptetünk.

54) **Hogyan határozzuk meg az LR(0) elemző automatájában a végállapotokat?**
closure read kanonikus halmazokat alakítunk ki, ha végállapot van, akkor redukálunk.

55) Mondd ki az LR(0)-elemzés nagy tételét!
Egy γ járható prefix hatására az elemző automatája a kezdőállapotból olyan állapotba kerül, amelyhez tartozó kanonikus halmaz éppen a γ járható prefixre érvényes LR(0) elemeket

tartalmazza.

56) **Milyen konfliktusok lehetnek az LR(0) elemző táblázatban?**
léptetés/redukálás konfliktus: az egyik elem léptetést, egy másik redukálást ír elő
redukálás/redukálás konfliktus:
az egyik elem az egyik szabály szerinti, a másik egy másik szabály szerinti redukciót ír elő
Az LR(0) tulajdonság biztosítja a táblázat konfliktusmentes kitöltését!

57) **Milyen esetben ír elo redukciót az SLR(1) elemzés?**
Ha az előreolvasott szimbólum benne van a szabályhoz tartozó nemterminális FOLLOW1 halmazában.

58) Hogy néz ki egy LR(1)-elem és mi a jelentése?
Ha $A \rightarrow \alpha$ a grammatika egy helyettesítési szabálya, akkor az $\alpha = \alpha 1\ \alpha 2$ tetszőleges felbontás és a terminális szimbólum (vagy a = #) esetén $[A \rightarrow \alpha 1\ .\alpha 2\ ,\ a]$ a grammatika egy LR(1)-eleme.
$A \rightarrow \alpha 1\ .\alpha 2$ az LR(1) elem magja, a pedig az előreolvasási szimbóluma.

59) Milyen esetben ír elo redukciót az LR(1) elemzés?
Ha az aktuális állapot **i**, és az előreolvasás eredménye az **a** szimbólum:

- ha $[A \rightarrow \alpha., a] \in Ii\ (A = S')$, akkor redukálni kell $A \rightarrow \alpha$ szabály szerint,

60) **Miért van általában lényegesen több állapota az LR(1) elemzőknek, mint az LR(0) (illetve SLR(1)) elemzőknek?**
Egy szabályt több jel is követhet. Mivel előreolvasunk mindig egy jelet, ha különbözőket olvasunk, az különböző állapotot jelent. Így ugyanaz az LR(0) illetve SLR(1) állapot többször is előfordul más előreolvasási szimbólummal.

61) **Mikor nevezünk két LR(1) kanonikus halmazt összevonhatónak?**
Ha a kanonikus halmaz párok csak az előreolvasási szimbólumokban különböznek.

62) Hogyan kapjuk meg az LALR(1) kanonikus halmazokat?
Az egyesíthető LR(1) kanonikus halmazokat vonjuk össze!

63) **Milyen fajta konfliktus keletkezhet a halmazok összevonása miatt az LALR(1) elemző készítése során?**
Redukálás-redukálás konfliktus.

64) **Milyen lépésekbol áll az LR elemzők hibaelfedo tevékenysége?**
- Hiba detektálása esetén meghívja a megfelelő hibarutint.
- A verem tetejéről addig töröl, amíg olyan állapotba nem kerül, ahol lehet az error szimbólummal lépni.
- A verembe lépteti az error szimbólumot.
- Az bemeneten addig ugorja át a soron következő terminálisokat, amíg a hibaalternatíva építését folytatni nem tudja.

65) Mi az if □ then □ else probléma?
A if (F) if (S) U else U részmondathoz több szintaxisfa is tartozik.
Nem egyértelmű a nyelvtan!
Problémát okoz mindegyik elemző esetén.

66) **Hogyan kell értelmezni a gyakorlatban az egymásba ágyazott elágazásokat, ha az az if □ then □ else probléma miatt nem egyértelmu?**
„Az else ág az őt közvetlenül megelőző if utasításhoz tartozik.”

67) **Hogyan oldják meg az if □ then □ else problémát az LR elemzők?**
Léptetéssel: Így az else az őt közvetlenül megelőző if utasításhoz fog tartozni.

68) Mire kell figyelni programozási nyelvek tervezésekor, ha el akarjuk kerülni az if □ then □ else problémát?
Vezessünk be if utasítás végét jelző kulcsszót.

69) **Milyen információkat tárolunk a szimbólumtáblában a szimbólumokról (fajtájuktól függetlenül)?**
Szimbólum neve,
Szimbólum attribútumai: definíció adatai, típus, tárgyprogram-beli cím, def. helye a forrásprogramban, szimbólumra hivatkozások a forrásprogramban.

70) Milyen információkat tárolunk a szimbólumtáblában a változókról?
típus
módosító kulcsszavak: const, static ...
címe a tárgyprogramban (függ a változó tárolási módjától)

71) Milyen információkat tárolunk a szimbólumtáblában a függvényekrol?
paraméterek típusa
visszatérési típus
módosítók
címe a tárgyprogramban

72) Milyen információkat tárolunk a szimbólumtáblában a típusokról?
egyszerű típusok: méret
rekord: mezők nevei és típusleírói
tömb: elem típusleírója, index típusleírója, méret
intervallum-típus: elem típusleírója, minimum, maximum
unio-típus: a lehetséges típusok leírói, méret

73) Milyen információkat tárolunk a szimbólumtáblában az osztályokról?
Hasonlatos, mint a típusoknál a rekord.
Mezők nevei és típusleírói.

74) **Mi a szimbólumtábla két alapveto művelete és mikor használja ezeket a fordítóprogram?**
keresés: szimbólum használatakor,
beszúrás:
- új szimbólum megjelenésekor
- tartalmaz egy keresést is: „Volt-e már deklarálva?”
-

75) Mi a változó hatóköre?
hatókör: „Ahol a deklaráció érvényben van.”

76) Mi a változó láthatósága?
láthatóság: „Ahol hivatkozni lehet rá a nevével.”
- része a hatókörnek
- az elfedés miatt lehet kisebb, mint a hatókör

77) Mi a változó élettartama?

„Amíg memóriaterület van hozzárendelve.”

78) Hogyan kezeljük változó hatókörét és láthatóságát szimbólumtáblával?

A szimbólumokat egy verembe tesszük.
Keresés:
- a verem tetejéről indul
- az első találatnál megáll

Blokk végén a hozzá tartozó szimbólumokat töröljük.

79) Milyen szerkezeteku szimbólumtáblákat ismersz?

verem-, faszerkezetű-, hash-szerezetű-

80) Miben tér el a névterek és blokkok kezelése a szimbólumtáblában?

A using direktíva használatakor az importált névtér szimbólumait be kell másolni a verembe (vagy legalább hivatkozást tenni a verembe erre a névtérre).

81) **Miért nem a szintaktikus elemző végzi el a szemantikus elemzés feladatait?**

A szintaktikus elemző környezetfüggetlen nyelvtannal dolgozik, míg a szemantikus környezetfüggővel, utóbbi több hibát kiszűr.

82) Mi a különbség a statikus és a dinamikus típusozás között?

Statikus: a kifejezésekhez fordítási időben a szemantikus elemzés rendel típust.
Dinamikus: a típusellenőrzés futási időben történik.

83) Mi a különbség a típusellenorzés és a típuslevezetés között?

Típusellenőrzés:
- minden típus a deklarációkban adott
- a kifejezések egyszerű szabályok alapján típusozhatók
- egyszerűbb fordítóprogram, gyorsabb fordítás
- kényelmetlenebb a programozónak

Típuslevezetés, típuskikövetkeztetés:
- a változók, függvények típusait (általában) nem kell megadni
- a típusokat fordítóprogram „találja ki” a definíciójuk,
- használatuk alapján
- bonyolultabb fordítóprogram, lassabb fordítás
- kényelmesebb a programozónak

84) **Mi a fordítóprogram teendője típuskonverzió esetén?**

A típusellenőrzés során át kell írni a kifejezés típusát ha szükséges, akkor a tárgykódba generálni kell a konverziót elvégző utasításokat.

85) Mik az akciószimbólumok?

A típusellenőrzés során át kell írni a kifejezés típusát ha szükséges, akkor a tárgykódba generálni kell a konverziót elvégző utasításokat.

86) Mik az attribútumok?

A nyelvtan szimbólumaihoz rendelt szemantikus típusok.

87) Hogyan kapnak értéket az attribútumok?

A szimbólumokhoz hozzárendeljük őket.

88) Mi a szintetizált attribútum?

A helyettesítési szabály bal oldalán áll abban a szabályban, amelyikhez az őt kiszámoló szemantikus rutin tartozik.

89) Mi a kitüntetett szintetizált attribútum?

Olyan attribútumok, amelyek terminális szimbólumokhoz tartoznak és kiszámításukhoz nem használunk fel más attribútumokat.

90) Mi az örökölt attribútum?
A helyettesítési szabály jobb oldalán áll abban a szabályban, amelyikhez az őt kiszámoló szemantikus rutin tartozik.

91) **Mivel egészítjük ki a nyelvtan szabályait attribútum fordítási grammatikák esetében?**
Megszorítással, hogy a kiértékelés egyszerűbb legyen:

⇒ particionált attribútum fordítási grammatikák

⇒ rendezett attribútum fordítási grammatikák

92) Mi a direkt attribútumfüggoség?
Ha az Y .b attribútumot kiszámoló szemantikus rutin használja az X .a attribútumot, akkor (X .a, Y .b) egy direkt attribútumfüggőség. Ezek a függőségek a függőségi gráfban ábrázolhatók.

93) Mi az S □ ATG? Milyen elemzésekhez illeszkedik?
Olyan attribútum fordítási grammatika, amelyben kizárólag szintetizált attribútumok vannak. A szemantikus információ a szintaxisfában a levelektől a gyökér felé terjed.
Jól illeszthető az alulról felfelé elemzésekhez!

94) Mi az L □ ATG? Milyen elemzésekhez illeszkedik?
Olyan attribútum fordítási grammatika, amelyben minden A → X1 X2 . . . Xn szabályban az attribútumértékek az alábbi sorrendben meghatározhatók:

A örökölt attribútumai
X1 örökölt attribútumai
X1 szintetizált attribútumai
X2 örökölt attribútumai
X2 szintetizált attribútumai
...
Xn örökölt attribútumai
Xn szintetizált attribútumai
A szintetizált attribútumai

Jól illeszkedik a felülről lefelé elemzésekhez.

95) Mi az assembly?
Programozási nyelvek egy csoportja.

96) Mi az assembler?
Az assembly programok fordítóprogramja.

97) **Milyen fobb regisztereket ismersz (általános célú, veremkezeléshez, adminisztratív célra)?**
eax, ebx,ecx,edx, esi, edi, ebp, esp,…

98) Mi köze van egymáshoz az eax, ax, al, ah regisztereknek?
eax 32 bitből áll: felső 16 bitnek nincs neve, alsó 16 bit: **ax**, ennek részei: felső bájt: **ah**, alsó bájt: **al**

99) Milyen aritmetikai utasításokat ismersz assemblyben?
inc, dec, add, sub, mul, div

100) **Mutasd be a logikai értékek egy lehetséges ábrázolását és a műveleteik megvalósítását assemblyben!**
and al,bl :Bitenkénti „és” művelet.
or eax,dword [C] :Bitenkénti „vagy” művelet.
xor word [D],bx :Bitenkénti „kizáró vagy” művelet.
not bl :Bitenkénti „nem” művelet.

101) Milyen feltételes ugró utasításokat ismersz?
je: „equal” - ugorj, ha egyenlő
jne: „not equal” - ugorj, ha nem egyenlő
jb: „below” - ugorj, ha kisebb ≡ jnae: „not above or equal” - nem nagyobb egyenlő
ja: „above” - ugorj, ha nagyobb ≡ jnbe: „not below or equal” - nem kisebb egyenlő
jnb: „not below” - nem kisebb ≡ jae: „above or equal” - nagyobb egyenlő
jna: „not above” - nem nagyobb ≡ jbe: „below or equal” - kisebb egyenlő
Ha előjeles egészekkel számolunk: jl („less”), jg („greater”), jnl, jng, jle, jge, . . .

102) Hogyan kapják meg a feltételes ugró utasítások a cmp utasítás eredményét?
A cmp eredményképpen egy flaget 0-ra vagy 1-re állít és azt vizsgálja meg az ugró utasítás.

103) **Milyen veremkezelo utasításokat ismersz assemblyben, és hogyan muködnek ezek?**
push eax: eax-et a verembe dobja. pop ebx: a verem tetején levőt ebx-be teszi.

104) **Melyik utasításokkal lehet alprogramot hívni és alprogramból visszatérni assemblyben?**
call: alprogram meghívása, ret: visszatérés

105) Hogyan generálunk kódot egyszeru típusok értékadásához?
a kifejezést az eax regiszterbe kiértékelő kód
mov [Változó],eax

106) Hogyan generálunk kódot egy ágú elágazáshoz?
a feltételt az al regiszterbe kiértékelő kód:
cmp al,1
je Then
jmp Vége
Then: a then-ág programjának kódja
Vége:

107) Hogyan generálunk kódot több ágú elágazáshoz?
az 1. feltétel kiértékelése az al regiszterbe
cmp al,1
jne near Feltétel_2
az 1. ág programjának kódja
jmp Vége . . .
Feltétel_n: az n-edik feltétel kiértékelése az al regiszterbe
cmp al,1
jne near Else
az n-edik ág programjának kódja
jmp Vége
Else: az else ág programjának kódja

Vége:

108) Hogyan generálunk kódot elöltesztelo ciklushoz?

Eleje: a ciklusfeltétel kiértékelése az al regiszterbe
cmp al,1
jne near Vége
a ciklusmag programjának kódja
jmp Eleje
Vége:

109) Hogyan generálunk kódot hátultesztelo ciklushoz?

Eleje: a ciklusmag programjának kódja
a ciklusfeltétel kiértékelése az al regiszterbe
cmp al,1
je near Eleje

110) **Hogyan generálunk kódot for ciklushoz?**

a „from" érték kiszámítása a [Változó] memóriahelyre
Eleje: a „to" érték kiszámítása az eax regiszterbe
cmp [Változó],eax
ja near Vége
a ciklusmag kódja
inc [Változó]
jmp Eleje
Vége:

111) **Hogyan generáljuk kezdoérték nélküli statikus változó definíciójának assembly kódját?**

section .bss
; a korábban definiált változók...
Lab12: resd 1 ; 1 x 4 bájtnyi terület

112) **Hogyan generáljuk kezdoértékkel rendelkezo statikus változó definíciójának assembly kódját?**

section .data
a korábban definiált változók...
Lab12: dd 5 ; 4 bájton tárolva az 5-ös érték

113) **Hogyan generáljuk aritmetikai kifejezés kiértékelésének assembly kódját? (konstans, változó, beépített függvény)**

konstans: mov eax,25
változó:mov eax,[X]
beépített függvény:
; a 2. kifejezés kiértékelése eax-be
push eax
; az 1. kifejezés kiértékelése eax-be
pop ebx
add eax,ebx

114) **Mutasd meg a különbséget a mindkét részkifejezést kiértékelo és a rövidzáras logikai operátorok assembly kódja között!**
(skip – ezt kikell keresni)

115) Hogyan generáljuk a goto utasítás assembly kódját?

goto Lab; => jmp Lab

116) **Miért nehéz a break utasítás kódgenerálását megoldani S □ATG használata esetén?**

A Vége címkét a ciklus feldolgozásakor generáljuk, pedig szükség van rá a break kódjában is! Azaz ez a címke egy örökölt attribútum...

117) Mit csinál a call és a ret utasítás?

A **call** Címke az eip regiszter tartalmát a verembe teszi, ez a call utáni utasítás címe, visszatérési címnek nevezzük.
Átadja a vezérlést a Címke címkéhez mint egy ugró utasítás
A **ret** kiveszi a verem legfelső négy bájtját és az eip regiszterbe teszi mint egy pop utasítás.
a program a veremben talált címnél folytatódik

118) **Hogyan adjuk át assemblyben az alprogramok paramétereit és hol lesz a lefutás után a visszatérési érték (C stílus esetén)?**

a paramétereket a verembe kell tenni a call utasítás előtt
C stílusú paraméterátadás: fordított sorrendben tesszük a verembe
- az utolsó kerül legalulra
- az első a verem tetejére

az ejárásból való visszatérés után a hívó állítja vissza a vermet és a visszatérési érték: az eax regiszterbe kerül

119) Hogyan épül fel az aktivációs rekord?

lokális változók <= esp
előző aktivációs rekord bázispointere <= ebp
visszatérési cím
1. paraméter
2. paraméter …….

120) Mi a bázismutató és melyik regisztert szoktuk erre a célra felhasználni?

Aktivációs rekordban az előző aktivációs rekordra mutató pointer, **ebp** regiszter.

121) Hol tároljuk alprogramok lokális változóit?

A verem tetején tároljuk őket.

122) **Mi a különbség az érték és a hivatkozás szerinti paraméterátadás assembly kódja között?**

érték szerint:
- a paraméterértékeket másoljuk a verembe
- ha az alprogram módosítja, az nem hat az átadott változóra

hivatkozás szerint
- az átadandó változóra mutató pointert kell a verembe tenni
- az alprogramban a lokális változó kiértékelése is módosul

123) **Milyen csoportokba oszthatók a változók tárolásuk szerint és a memória mely részeiben tároljuk az egyes csoportokba tartozó változókat?**

statikus memóriakezelés:
- a .data vagy .bss szakaszban
- globális vagy statikusnak deklarált változók
- előre ismerni kell a változók méretét, darabszámát

dinamikus memóriakezelés
- blokk-szerkezethez kötődő, lokális változók: verem
- tetszőleges élettartamú változók: heap memória

124) Mi a különbség a lokális és a globális optimalizálás között?

lokális: kis programrészletek átalakítása
globális: a teljes program szerkezetét kell vizsgálni

125) Mit jelent a gépfüggo optimalizálás?

gépfüggő optimalizálás: az adott architektúra sajátosságait használja ki

126) Mit nevezünk alapblokknak és melyik optimalizálás során van szerepe?
Egy programban egymást követő utasítások sorozatát alapblokknak nevezzük, ha:
- az első utasítás kivételével egyik utasítására sem lehet távolról átadni a vezérlést
- az utolsó utasítás kivételével nincs benne vezérlés-átadó utasítás
- az utasítás-sorozat nem bővíthető a fenti két szabály megsértése nélkül

Lokális optimalizálás során van szerepe: egy alapblokkon belüli átalakításnál.

127) Mutass példát konstansok összevonására!
a:= 1+b+3+4; helyett a:= b+8;

128) Mutass példát konstans továbbterjesztésére!

a:= 6;	helyett	a:=6;
b:= a/2;		b:=3;
c:=b+5;		c:=8;

129) Mutass példát azonos kifejezések többszöri kiszámításának elkerülésére!

x := 20 - (a * b);	helyett	t:=a*b;
y := (a * b) ^ 2;		x:=20-t;
		y:=t^2;

130) Mi az ablakoptimalizálási technika lényege?
egyszerre csak egy néhány utasításnyi részt vizsgálunk a kódból , majd a vizsgált részt előre megadott mintákkal hasonlítjuk össze. Ha illeszkedik, akkor a mintához megadott szabály szerint átalakítjuk és ezt az „ablakot" végigcsúsztatjuk a programon.

131) **Mi a különbség a kódkiemelés és a kódsüllyesztés között?**
Kódkiemelésnél a blokk elé visszük ki a változót kódsüllyesztésnél mögé.

132) Hogyan változtatja a program sebességét és méretét a ciklusok kifejtése?
Mérlegelni kell, hogy a méret és a sebesség mennyire fontos...

133) Mi a frekvenciaredukálás lényege?
Költséges utasítások „átköltöztetése" ritkábban végrehajtódó alapblokkba.

134) Mutass példát eros redukcióra!

Helyett

```
for( int i=a; i<b; i+=c ){
cout << 3*i;
}
```

```
int t1 = 3*a;
int  t2 = 3*c;
for( int i=a; i<b; i+=c ){
cout << t1; t1 +=t2;
}
```

by **Varga Mátyás** (VAMQAAI.ELTE)
2011.01.12.
Megjegyzés: A várható kérdések ***javarészét*** lefedi, ha tudod ezeket, jó eséllyel meglesz a beugró, de előfordulhat pár másik kérdés is. 2010-2011-1 félévnek előadás diái alapján kidolgozva Dévai Gergelynél.