1. Primjer SISD raÄŤunala je:
(a) raÄŤunalo temeljeno na viĹˇejezgrenom procesoru
(b) Von Neumannovo raÄŤunalo
(c) vektorski procesor na grafiÄŤkoj kartici
(d) redundantno raÄŤunalo u kojem viĹˇe izvrĹˇnih jedinica obraÄ‘uje iste podatke
(e) paralelno zbrajalo

2. Primjer SIMD raÄŤunala je:
(a) raÄŤunalo temeljeno na viĹˇejezgrenom procesoru
(b) Von Neumannovo raÄŤunalo
(c) vektorski procesor na grafiÄŤkoj kartici
(d) redundantno raÄŤunalo u kojem viĹˇe izvrĹˇnih jedinica obraÄ‘uje iste podatke
(e)- paralelno zbrajalo

3. TipiÄŤni skalarni RISC procesor ima:
(a) jednoadresne aritmetiÄŤke instrukcije
(b) aritmetiÄŤke instrukcije s memorijskim operandima
(c) akumulatorsku arhitekturu
(d) troadresne aritmetiÄŤke instrukcije bez memorijskih operanda
(e) tablicu meÄ‘uovisnosti (scoreboard)

4. Koncept upravljanja tokom podataka koristi se u:
(a) originalnom Von Neumannovom modelu
(b) protoÄŤnim raÄŤunalima
(c) CISC raÄŤunalima
(d) superskalarnim raÄŤunalima
(e) viĹˇejezgrenim raÄŤunalima

5. Za superskalarne RISC arhitekture je specifiÄŤno
da se usporedno izvoÄ‘enje slijednog programa
pospjeĹˇuje prvenstveno:
(a) dubokom protoÄŤnom strukturom
(b) dinamiÄŤkim rasporeÄ‘ivanjem instrukcija u sklopovlju procesora
(c) malom ali brzom priruÄŤnom memorijom
(d) adresnim preslikavanjem
(e) statiÄŤkim rasporeÄ‘ivanjem instrukcija tijekom prevoÄ‘enja

6. Moderni superskalarni procesori tipiÄŤno postiĹľu:
(a) CPI > 3 GHz
(b) CPI > 100 MHz
(c) CPI > 100
(d) CPI $\in [2,10]$
(e) CPI < 1

7. Procesori 8086 i Core i7 920 imaju:
(a) srodnu instrukcijsku arhitekturu, ali razliÄŤitu organizaciju
(b) srodnu organizaciju, ali razliÄŤitu instrukcijsku arhitekturu
(c) srodnu instrukcijsku arhitekturu i srodnu organizaciju
(d) isti broj vanjskih izvoda (pinova)
(e) kompatibilnu adresnu sabirnicu

8. VeÄ‡ina instrukcija arhitekture x86 podrĹľava:
(a) 0 memorijskih operanada
(b) 1 memorijski operand
(c) 2 memorijska operanda
(d) 3 memorijska operanda
(e) 4 memorijska operanda

9. Podatkovna sekcija objektnog modula tipiÄŤno sadrĹľi i:

(a) program u strojnom k\^ odu
(b) lokalne varijable ÄŤije vrijednosti se gube nakon izlaska iz procedure
(c) statiÄŤke varijable (lokalne i globalne)
(d) dinamiÄŤki alociranu memoriju (malloc, new)
(e) program u izvornom k\^ odu

10. Relokacijske informacije objektnog modula sadrĹľe popis:
(a) uputa za izmjenu sekcija modula pri odreÄ‘ivanju njihovog konaÄŤnog poloĹľaja
(b) identifikatora koje modul definira ili referencira
(c) statiÄŤkih varijabli koje se koriste u modulu
(d) lokalnih varijabli modula
(e) sekcija objektnog modula

11. Objektni moduli programskog jezika C na arhitekturi x86
tipiÄŤno predviÄ‘aju relociranje
sljedeÄ‡ih elemenata programske sekcije:
(a) svih instrukcija grananja
(b) svih instrukcija uvjetnog grananja
(c) svih instrukcija grananja na potprograme
(d) svih instrukcija grananja na potprograme izvan modula
(e) svih instrukcija uvjetnog grananja i grananja na potprograme

12. Najbolja performansa interpretiranih programa (Java, Python, ...)
tipiÄŤno se postiĹľe optimiranjem:
(a) tijekom izvrĹ¡avanja programa
(b) tijekom prevoÄ‘enja u bajtni meÄ‘ukod
(c) tijekom povezivanja objektnih modula u izvrĹ¡ivu datoteku
(d) virtualnog stroja
(e) prevoditelja

13. ProtoÄŤna arhitektura MIPS
u svakom ciklusu signala takta izvrĹ¡i:
(a) barem dva memorijska pristupa
(b) najviĹ¡e dva memorijska pristupa
(c) toÄŤno dva memorijska pristupa
(d) najviĹ¡e jedan memorijski pristup
(e) toÄŤno jedan memorijski pristup

14. U protoÄŤnom raÄŤunalu sa zajedniÄŤkom jednoadresnom priruÄŤnom memorijom podataka i instrukcija
naroÄŤito moĹľemo oÄŤekivati:
(a) podatkovne hazarde
(b) strukturne hazarde
(c) upravljaÄŤke hazarde
(d) oteĹľano izvoÄ‘enje samomodificirajuÄ‡eg koda
(e) ubrzanje od 20\%

15. Kakvo prosljeÄ‘ivanje se tipiÄŤno koristi
za smanjenje latencije instrukcije grananja
(i oznaÄŤava redni broj instrukcije)?
(a) ID[i] $\rightarrow$ IF[i+2] (na slajdovim piĹ¡e ID[i] $\rightarrow$ IF[i+1]!!!)
(b) EX[i] $\rightarrow$ ID[i+1]
(c) [i+1] $\rightarrow$ IF[i]
(d) ME[i+1] $\rightarrow$ ID[i]
(e) WB[i] $\rightarrow$ ID[i+1]

16. Koncept protoÄŤnosti je koristan jer omoguÄ‡ava:
(a) istu performansu uz manji broj tranzistora
(b) iskoriĹ¡tavanje instrukcijskog paralelizma

(c) CISC arhitekturama da se po performansi izjednaÄŤe s RISC-om
(d) smanjivanje potrebnog broja registara
(e) ublaĹľavanje resursnih konflikata