Principi programiranja. Ogledni primjeri pitanja za Završni ispit (40 poena). Od naslova 21. Primjer mikroprocesora: i8086 do naslova 46. Naknadno dodato: jednostavni primjeri za ulaz–izlaz. Doći će dva pitanja – teorija i dva pitanja – programi na jeziku asemblera za Intel 8086.

1. Glavni registri procesora Intel 8086 (AH, ..., IP), obrazovanje fizičke adrese po formuli $a = 16s + d$.

2. Registar PSW procesora Intel 8086 tj. flagovi.

3. O programu debug.exe i njegovim komandama A, G, D, T i U (assemble, go, dump, trace i unassemble).

4. Smisao naredbi MOV, PUSH, POP, IN, OUT, PUSHF i POPF (služe za prenos podataka, Intel 8086).

5. Smisao naredbi ADD, ADC, SUB, SBB, CMP, INC i DEC (aritmetičke naredbe, Intel 8086).

6. Smisao naredbi MUL, IMUL, DIV i IDIV (aritmetičke naredbe, Intel 8086).

7. Smisao naredbi JMP, JL, JLE, JG, JGE i JCXZ (uslovni i bezuslovni skokovi, Intel 8086).

8. Smisao naredbi CALL, RET, RETF, INT, INTO i IRET (potprogrami i prekidi, Intel 8086).

9. Metode adresiranja: immediate (kada piše n), direct (kada piše $[n]$), register (kada piše AX ili ...) i indirect (kada piše [BX] ili [BP] ili [SI] ili [DI]).

10. Asemblerske direktive DB immed i DW immed (kod debug.exe). Define byte, define word.

11. Sistem prekida procesora Intel 8086: samo o prekidima od $n = 0$ do $n = 4$ i o tri moguća izvora–uzroka da dođe do prekida.

12. Neke funkcije sistemskog prekida 21H: samo funkcija AH = 1, AH = 2, AH = 8 i AH = 9.

13. Pregled BIOS–ovih i DOS–ovih prekida: samo o INT 5 print screen, INT 10 (funkcija AH = 2) set cursor position i INT 20 terminate program.

14. Jedan konkretan primjer rutine za obradu prekida: o kodu rutine INT 5 (print screen) u glavnim crtama.

15. O mikroprocesoru Intel 80286.

16. O mikroprocesoru Intel 80386.

17. O mikroprocesoru Pentium: četiri generacije Pentiuma, tekuća traka (pipeline) i dohvatanje unaprijed (prefetch).

18. Temeljni pojmovi o operativnim sistemima: definicija, struktura i jezik.

19. O podjeli operativnih sistema na generacije (iz naslova Kratak pregled razvoja operativnih sistema).

20. Operativni sistemi DOS i Windows (iz naslova Kratak pregled razvoja operativnih sistema).

21. Pojam liste, operacije INSERT i DELETE i kako se lista realizuje (deklariše) u Paskalu [ne treba program].

22. O pokazivačima u Paskalu: pojam pokazivača i potprogrami NEW i DISPOSE.

23. Pojam steka, operacije sa stekom i kako se stek realizuje [ne treba program].

24. Pojam reda, operacije sa redom i predstavljanje u memoriji (sa kružnim svojstvom).

25. Pojam grafa i zadatak o Ojlerovom ciklusu.

26. Pojam drveta i binarnog drveta. Kako se binarno drvo realizuje (deklariše) u Paskalu [ne treba program].

27. Jednostavni primjer za ulaz–izlaz: rad sa ekranom (display).

1. Sastaviti program na jeziku asemblera (za procesor Intel 8086) za računanje četiri broja: $y_1 = 2ab + 1$, $y_2 = y_1 - 2$, $y_3 = y_2 - 2$ i $y_4 = y_3 - 16$, gdje su a i b dati brojevi. Svi brojevi su veličine po dva bajta. Nisu potrebne naredbe za ulaz odnosno izlaz, već samo predvidjeti adrese za brojeve.

2. Sastaviti program na jeziku asemblera u kome se računaju cijeli brojevi $y_1 = -16a - 20b - 24c$, $y_2 = a^2 + ab + ac$ i $y_3 = \frac{y_1^2+1}{y_2^2+1}$, gdje su a, b i c dati brojevi.

3. Vježba sa stekom. Upisati u stek redom brojeve (po dva bajta) $x_1 = 11$, $x_2 = 12$, $x_3 = 13$ i $x_4 = 14$ (ovo je u dekadnom sistemu), a onda izračunati (čitanjem sa steka) $y = x_1 + 2x_2 + 3x_3 + 4x_4$.

4. Vježba sa stekom. Upisati u stek redom brojeve (po dva bajta) $x_1 = 11$, $x_2 = 12$, $x_3 = 13$ i $x_4 = 14$ (ovo je u dekadnom sistemu), a onda izračunati (čitanjem sa steka) $y = x_1^2 + x_2^2$.

5. Učitati preko tastature četiri slova (upisati ih u memoriju), prekid INT 21, funkcija AH = 1 ili svejedno AH = 8 (ovo je u heksadekadnom). Zatim štampati u prvom redu učitana slova, a u drugom redu ta ista slova, prekid INT 21, funkcija AH = 2, Carriage return + Line feed = prelazak na početak novog reda. Sastaviti odgovarajući program na jeziku asemblera.

6. Učitati preko tastature četiri slova, a zatim dvaput štampati učitanu četvorku slova pomoću prekida INT 21, funkcija AH = 9 čiji je smisao: štampanje niza slova (prikazivanje niza slova na ekran). Sastaviti odgovarajući program na jeziku asemblera.

7. Sastaviti program na jeziku asemblera za računanje vrijednosti izraza $y = \{\ 2a+2b$ ako je $c < 0$, $8a + 8b$ ako je $c \geq 0$. Uputstvo: na primjer, koristiti kombinaciju naredbi CMP i JGE.

8. Sastaviti program na jeziku asemblera za računanje $y_1 = a^2 + b^2 - 100$ i $y_2 = \{\ -1$ ako je $y_1 < 0$, 0 ako je $y_1 = 0$, 1 ako je $y_1 > 0$.

9. Program na asembleru za $y = \sum_{k=a}^{b}(k+1)$, pomoću petlje ili pomoću uslovnih skokova, tj. ne koristiti formulu za $1 + \ldots + n$.

10. Program na asembleru za $y = \sum_{k=a}^{b}(-1)^k k$, pomoću petlje ili pomoću uslovnih skokova, tj. ne koristiti neku gotovu formulu.

11. Rad sa nizom (indirektno adresiranje). Treba izračunati y, gdje je y zbir 16 brojeva $x_1, \ldots, x_{16}$ (po dva bajta) koji se nalaze na adresama od 1000 do 101F u tekućem segmentu.

12. Rad sa nizom (indirektno adresiranje). Treba izračunati y, gdje je y najveći među 16 brojeva (po dva bajta) koji se nalaze na adresama od 1000 do 101F u tekućem segmentu.

Teorijska pitanja će doći kako ovdje piše, a zadaci (treće i četvrto pitanje) će doći isti ili slični.

U posljednjem pitanju (četvrtom) dozvoljeno je da se koriste simboličke labele, kod skokova, kao što je uobičajeno u mnogim asemblerima. Primjer: JMP lab. Ili npr. da umjesto JMP 12C itd. 12C: XOR AX, AX pišemo JMP tamo itd. tamo: XOR AX, AX. Simbolička adresa može da bude "lab" ili "tamo" ili "mjesto" ili neka druga riječ (bilo koja riječ), bilo koja labela.

U nastavku dajemo dva najprostija moguća primjera iz asemblera za procesor Pentium ili slično pod operativnim sistemom Windows. U postavkana zadataka brojevi su prikazani u dekadnom sistemu, a u rješenjima u heksadekadnom. Svi brojevi su veličine po dva bajta.

1. Napisati program na jeziku asemblera za računanje $y_1 = 2a - 5$ i $y_2 = -(b+5)$. Vrijednosti ulaznih podataka $a = 18$ i $b = 14$ treba upisati u registre AX odnosno BX pomoću naredbi MOV. Na kraju, rezultate y_1 i y_2 treba ostaviti u registrima CX odnosno DX.

Slijedi rješenje, a posebno je prikazan sadržaj AX, BX, CX i DX tokom izvršavanja korak po korak.

	unosimo naredbe		AX	BX	CX	DX
100	`MOV AX, 12`	AX ← 12	12	0	0	0
103	`MOV BX, E`	BX ← E	12	E	0	0
106	`SHL AX, 1`	AX ← 2 · AX	24	E	0	0
108	`SUB AX, 5`	AX ← AX − 5	1F	E	0	0
10B	`MOV CX, AX`	CX ← AX	1F	E	1F	0
10D	`ADD BX, 5`	BX ← BX + 5	1F	13	1F	0
110	`NEG BX`	BX ← −BX	1F	FFED	1F	0
112	`MOV DX, BX`	DX ← BX	1F	FFED	1F	FFED
114	`INT 20`	stop	1F	FFED	1F	FFED

2. Napisati program na jeziku asemblera za računanje $y_1 = a + b + 1$, $y_2 = a - b - 1$, $y_3 = y_1 \cdot y_2$ i $y_4 = y_1/y_2$. Našim promjenljivim a i b treba dodijeliti vrijednosti $a = 16$ i $b = 12$ pomoću direktiva. Treba ostaviti rezultate $y_1, \ldots, y_4$ na adresama 204, 206, 208 i 20A (hex).

Slijedi rješenje, a posebno je prikazan sadržaj registara AX i BX tokom izvršavanja korak po korak.

unosimo podatke			u memoriji:	
200	`DW 10`	define word (two bytes), $a = 10$	200, 201	10
202	`DW C`	define word (two bytes), $b = \mathrm{C}$	202, 203	C

unosimo naredbe			u procesoru:
100	`MOV AX, [200]`	$\mathrm{AX} \leftarrow c(200)$	$\mathrm{AX} = 10, \mathrm{BX} = 0$
103	`ADD AX, [202]`	$\mathrm{AX} \leftarrow \mathrm{AX} + c(202)$	$\mathrm{AX} = 1\mathrm{C}, \mathrm{BX} = 0$
107	`INC AX`	$\mathrm{AX} \leftarrow \mathrm{AX} + 1$	$\mathrm{AX} = 1\mathrm{D}, \mathrm{BX} = 0$
108	`MOV [204], AX`	$c(204) \leftarrow \mathrm{AX}$	$\mathrm{AX} = 1\mathrm{D}, \mathrm{BX} = 0$
10B	`MOV AX, [200]`	$\mathrm{AX} \leftarrow c(200)$	$\mathrm{AX} = 10, \mathrm{BX} = 0$
10E	`SUB AX, [202]`	$\mathrm{AX} \leftarrow \mathrm{AX} - c(202)$	$\mathrm{AX} = 4, \mathrm{BX} = 0$
112	`DEC AX`	$\mathrm{AX} \leftarrow \mathrm{AX} - 1$	$\mathrm{AX} = 3, \mathrm{BX} = 0$
113	`MOV [206], AX`	$c(206) \leftarrow \mathrm{AX}$	$\mathrm{AX} = 3, \mathrm{BX} = 0$
116	`MOV AX, [204]`	$\mathrm{AX} \leftarrow c(204)$	$\mathrm{AX} = 1\mathrm{D}, \mathrm{BX} = 0$
119	`MOV BX, [206]`	$\mathrm{BX} \leftarrow c(206)$	$\mathrm{AX} = 1\mathrm{D}, \mathrm{BX} = 3$
11D	`MUL BX`	$\mathrm{AX} \leftarrow \mathrm{AX} \cdot \mathrm{BX}$	$\mathrm{AX} = 57, \mathrm{BX} = 3$
11F	`MOV [208], AX`	$c(208) \leftarrow \mathrm{AX}$	$\mathrm{AX} = 57, \mathrm{BX} = 3$
122	`XOR DX, DX`	$\mathrm{DX} \leftarrow 0$	$\mathrm{AX} = 57, \mathrm{BX} = 3$
124	`MOV AX, [204]`	$\mathrm{AX} \leftarrow c(204)$	$\mathrm{AX} = 1\mathrm{D}, \mathrm{BX} = 3$
127	`MOV BX, [206]`	$\mathrm{BX} \leftarrow c(206)$	$\mathrm{AX} = 1\mathrm{D}, \mathrm{BX} = 3$
12B	`DIV BX`	$\mathrm{AX} \leftarrow \mathrm{AX}/\mathrm{BX}$	$\mathrm{AX} = 9, \mathrm{BX} = 3$
12D	`MOV [20A], AX`	$c(20\mathrm{A}) \leftarrow \mathrm{AX}$	$\mathrm{AX} = 9, \mathrm{BX} = 3$
130	`INT 20`	stop	$\mathrm{AX} = 9, \mathrm{BX} = 3$

čitamo pomoću `D 204 20B` (dump)	u memoriji:	
$y_1 = c(204) = 1\mathrm{D}$ two bytes	204, 205	1D
$y_2 = c(206) = 3$ two bytes	206, 207	3
$y_3 = c(208) = 57$ two bytes	208, 209	57
$y_4 = c(20\mathrm{A}) = 9$ two bytes	20A, 20B	9

Isprobati pomoću nekog programa za propuštanje asemblerskih vježbi.

Na primjer, pomoću programa emu8086. Treba preuzeti program sa adrese www.emu8086.com. Npr. piše: Study computer architecture and assembly language programming. Ili ga preuzeti sa nekog drugog sajta. Dobiće se three months free trial.