Αρχιτεκτονική Η/Υ

Ασκήσεις Επανάληψης και Εξετάσεων

1. Με τη βοήθεια του διαγράμματος λειτουργίας CPU-Μνήμης (από την εισαγωγή) εξηγείστε το κύκλο Ανάκλησης – Εκτέλεσης. Γράψτε τα βήματα σε μορφή ψευδοκώδικα. Αναφέρατε τους βασικούς καταχωρητές της CPU και τη λειτουργία τους.

2. Εξηγείστε τους όρους Διερμηνεία και Μικροκώδικας στα πλαίσια της αρχιτεκτονικής υπολογιστών. Εξηγείστε τα υπέρ και τα κατά της προσέγγισης αυτής καθώς και τις διαφορές RISC και CISC. Ποιές είναι οι αρχές κατασκευής των σύγχρονων επεξεργαστών;

3. Αναπτύξτε σύντομα τα βήματα Εξυπηρέτησης Διακοπής. Πώς τροποποιείται ο κύκλος Ανάκλησης – Εκτέλεσης; Ποιος καταχωρητής χρησιμοποιείται; Τι άλλο υλικό απαιτείται συνήθως; Ποια η διαφορά περιόδευσης (Polling) και Ανύσματος Διακοπής (Interrupt Vector); Πώς υλοποιούνται τα επίπεδα διακοπών;

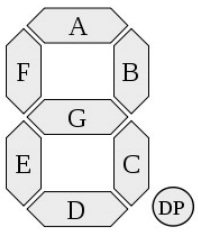
4. Εξηγείστε τις διαφορές των αρχιτεκτονικών Συσσωρευτή (Accumulator), Στοίβας (Stack) και Γενικών Καταχωρητών (Register Bank) στο υλικό. Ποιες διαφορές έχουμε στο επίπεδο του Συνόλου Εντολών; Εξηγείστε με ένα παράδειγμα πχ πρόσθεσης δύο τιμών μεταβλητών.

5. Σχεδιάστε ένα κύκλωμα περιττής ισοτιμίας για το βασικό κώδικα ASCII (7 bits).

6. Σχεδιάστε ένα κύκλωμα που παράγει κώδικα Hamming για μια λέξη των 16 bits με χρήση άρτιας ισοτιμίας.

7. Σχεδιάστε ένα κύκλωμα που να μετατρέπει έναν αριθμό των 4-bits από το σύστημα του Συμπληρώματος του 1 στο σύστημα Πρόσημο – Μέγεθος. Απλοποιήστε τις συναρτήσεις και χρησιμοποιείστε μόνο πύλες NAND δύο εισόδων.

8. Απλή εφαρμογή οθόνης επτά τμημάτων: σχεδιάστε ένα κύκλωμα που να δέχεται ως είσοδο τη δυαδική αναπαράσταση ενός αριθμού (από 0 έως 9) και εμφανίζει το αποτέλεσμα στη μορφή (δες http://en.wikipedia.org/wiki/Seven-segment_display_character_representations)



   Παραλείψτε την υποδιαστολή (DP). Επεκτείνετε το σύστημά σας για εμφάνιση αριθμού στο Συμπλήρωμα του 2 (το DP θα μπορούσε να είναι πρόσημο).

9. Με βάση το παραπάνω σύνδεσμο της Wikipedia αναπτύξτε ένα σύστημα επτά σημείων για

αναπαράσταση απλού κώδικα ASCII (κεφαλαία Αγγλικά και αριθμούς 0 έως 9).

10. Στο σύστημα 4 προσαρμόστε ένα απλό πληκτρολόγιο. Έστω οτι έχετε στη διάθεσή σας πλήκτρα για γράμματα και αριθμούς, αντί για αναπαραστάσεις ASCII.

11. Αναλύστε το κύκλωμα του Συγκριτή με χρήση μόνο πυλών NAND δύο εισόδων. Πόσα transistors απαιτούνται;

12. Επεκτείνετε το κύκλωμα του Ολισθητή ώστε να επιτρέπει απλή διέλευση και Αριθμητική Ολίσθηση, δηλαδή να λαμβάνει υπ' 'οψη το πρόσημο κατά την ολίσθηση.

13. Επεκτείνετε το κύκλωμα του Ολισθητή ώστε να επιτρέπει και Περιστροφή (Rotate) αριστερά/δεξιά. Απαιτείται έλεγχος εισόδου και εξόδου.

14. Σχεδιάστε ένα κύκλωμα ελέγχου Υπερχείλισης (Overflow) και ένα κύκλωμα που να ελέγχει αν το αποτέλεσμα είναι Μηδέν για τον Αθροιστή.

15. Σχεδιάστε και απλοποιείστε πλήρως έναν Αφαιρέτη (μόνο με πύλες NAND δύο εισόδων). Ποια είναι η σχέση του κυκλώματος με αυτό του Αθροιστή; Πώς θα μπορούσαν οι δύο πράξεις να εκτελούνται από ένα κύκλωμα (με Επιλογή Συνάρτησης);

16. Να γραφούν σε πλήρη μορφή JAS και να εκτελεστούν τα προγράμματα ελέγχου του προσομοιωτή Mic-1 (δες σχετικό εγχειρίδιο χρήσης).

17. Μετατρέψτε τα απλά templates των σημειώσεων από C/Java σε πλήρη προγράμματα JAS. Σε κάθε πρόγραμμα μελετήστε τε το κώδικα IJVM που αντιστοιχεί. Κατά την εκτέλεση του κάθε προγράμματος σε IJVM mode παρατηρείστε τις περιοχές μνήμης (κυρίως την εξέλιξη της στοίβας). Τελικά, παρατηρείστε τις διαδρομές δεδομένων, την ALU και σχολιάστε τις τιμές των καταχωρητών PC, MAR, MBR, LV και SP. Πως προκύπτουν;

18. Γράψτε τον κώδικα JAS που ακολουθεί σε IJVM (δηλαδή δεκαεξαδική μορφή) αλλά και σε Java. Ποια θα είναι η τιμή του x στο τέλος της εκτέλεσης;

```
        BIPUSH 3
        ISTORE i
        BIPUSH 1
        ISTORE x
L1:     ILOAD i
        IFLT L2
        ILOAD x
        DUP
        IADD
        ISTORE x
        IINC i -1
        GOTO L1
L2:     ...
```

19. Το ίδιο όπως στην άσκηση 1. Τροποποιήστε τη συνθήκη ώστε να χρησιμοποιεί έλεγχο ισότητας αντί για ανισότητα.

```
        BIPUSH 0
        ISTORE x
        BIPUSH 0
        ISTORE i
L1:     BIPUSH 19
        ILOAD i
```

```
        ISUB
        IFLT L2
        ILOAD x
        ILOAD i
        IADD
        ISTORE x
        IINC i 1
        GOTO L1
L2:     HALT
```

20. Έστω οτι αρχικά ο SP δείχνει στη τιμή

```
11111111111111111111111111111101 = -3
```

Η περιοχή των μεθόδων περιέχει τις παρακάτω εντολές και ο PC δείχνει στη πρώτη θέση. Τι θα περιέχει η στοίβα μετά την εκτέλεση του προγράμματος;

```
01011001
10011011
00000000
00000110
10100111
00000000
00000111
00010000
00000000
01011111
01100100
```

21. Γράψτε το πρόγραμμα JAS που αντιστοιχεί στη παρακάτω δεκαεξαδική ακολουθία. Στη συνέχεια γράψτε το πρόγραμμα Java (ή C) που αντιστοιχεί στο πρόγραμμα JAS.

```
1004
3600
1005
3601
1001
3602
1000
3603
1501
1502
64
9b0010
1500
1503
60
3603
840201
a7ffee
00
```

22. Μεταγλωττίστε το απόσπασμα Java/C που ακολουθεί σε JAS και στη συνέχεια σε IJVM.

```
int i, k;

i=3;
k=0;
while (i>0){
  k=k+5;
  i=i-1;
```

```
}
```

23. Γράψτε το πρόγραμμα JAS που ακολουθεί σε IJVM. Καταγράψτε τη κατάσταση της στοίβας μετά την εκτέλεση κάθε εντολής (για δύο επαναλήψεις).

```
// Display all printable ASCII characters
.constant
            one 1
            start 32
            stop 126
.end-constant
.main
            LDC_W start       // Load on top of stack constant start
next:       DUP               // Duplicate top of stack:keep this for counter
            OUT               // Display top of stack
            DUP               // Duplicate top of stack for subtraction
            LDC_W stop        // Load on stack constant stop
            ISUB              // Pop two values, subtract, push on top of stack
            IFEQ done         // If stop is reached go to done
            LDC_W one         // Load on top of stack one
            IADD              // Pop two values, add and push on top of stack
            GOTO next         // Loop again
done:       POP               // Empty stack from counter
            HALT              // End
.end-main
```

24. Γράψτε ένα πρόγραμμα που να υπολογίζει το γινόμενο δύο θετικών ακεραίων του ενός byte. Υπάρχουν τουλάχιστο δύο μέθοδοι υλοποίησης: επαναληπτικές προσθέσεις (η εύκολη μέθοδος) και ο αλγόριθμος ολίσθησης-πρόσθεσης (δίνεται παρακάτω).

    Δεδομένων δύο αριθμών a, b υπολογίζουμε τα x = a * b.

```
r=0;  mask=0x01;
for (i = 0; i < 8; i++) { // for 8 bit positive number
   if ((a & mask) != 0)   // bit-wise AND
      r = r + b;
   b = b << 1;
   mask = mask << 1;
}
```

25. Γράψτε ένα πρόγραμμα ακέραιας διαίρεσης δύο θετικών ακεραίων αριθμών (εύρεση πηλίκου και υπολοίπου). Δεδομένων δύο αριθμών a >= b υπολογίζουμε τα x, y ώστε a = b * x + y.

```
x=0; c=a;
while (c >= b) {
      x=x+1;  c=c-b;
}
y=c;
```

26. Έστω μια έκφραση της μορφής x = x + y – z + w - 3. Γράψτε ένα πρόγραμμα JAS που να δίνει αρχικές τιμές στα x, y, z, να υπολογίζει τη τιμή του x και αποθηκεύει το αποτέλεσμα.

27. Γράψτε ένα πρόγραμμα JAS που να δέχεται τρεις αριθμούς στο διάστημα 0 έως 120 (γωνίες τριγώνου) και να παράγει ως αποτέλεσμα 0 αν δεν είναι τρίγωνο, 1 αν είναι ισόπλευρο, 2 αν είναι ισοσκελές, 3 αν είναι σκαλινό.

28. Γράψτε και εκτελέστε το παράδειγμα απλής κλήσης μεθόδου των σημειώσεων. Ελέγξτε την περιοχή μεθόδων για να δείτε την τοποθέτηση της μεθόδου και την IJVM κωδικοποίηση της κλήσης και της επιστροφής. Μελετήστε την εκτέλεση του προγράμματος ελέγχοντας τη κατάσταση της στοίβας και των καταχωρητών πριν και μετά τη κλήση της μεθόδου, καθώς και πριν και μετά την επιστροφή από τη μέθοδο.

29. Με βάση το Σύνολο Εντολών IJVM/JAS που δίνεται στο Διάγραμμα 1.1 και το απόσπασμα κώδικα C που δίνεται Διάγραμμα 1.2. ζητούνται τα εξής: Γράψτε το αντίστοιχο πρόγραμμα σε συμβολική γλώσσα JAS. Σχεδιάστε το μοντέλο μνήμης κατά την έναρξη εκτέλεσης του προγράμματος και δείξτε ποιοι καταχωρητές ελέγχουν τις αντίστοιχες περιοχές μνήμης. Δώστε τα περιεχόμενα της στοίβας πριν και μετά την κλήση της μεθόδου, καθώς και πριν και μετά την επιστροφή από τη μέθοδο.

```
// ***** OBJREF
.constant

      objref 0xCAFE
.end-constant
// ****** main ( ) {
.main
// ****** int x, y, z;
.var
.end-var
// ****** x = 10;
// ****** y = 11;
// ****** z = my_add (x, y);
      HALT
// ****** }
.end-main
// ***** int my_add (int p1, int p2) {
.method my_add (p1, p2)
// ***** int temp;
.var
      temp
.end-var
// ***** temp = p2 + p1
// ***** return temp
// ***** };
.end-method
```

30. Να γραφούν απλές μέθοδοι για την εισαγωγή και εμφάνιση τοπικής μεταβλητής, του τύπου get(char x) και put(char x).

31. Να γραφούν απλές μέθοδοι της μορφής int max (int x, int y) και int min (int x, int y).

32. Να γραφούν απλές μέθοδοι της μορφής (να μη ληφθούν υπ' όψη φαινόμενα υπερχείλισης)
    int mul (int x, int y)
    int div (int x, int y)
    int mod (int x, int y)
    int abs (int x)

33. Να γραφεί μια απλή αναδρομική μέθοδος (πχ υπολογισμός αθροίσματος 1+2+..+N).

34. Να γραφεί μια αναδρομική μέθοδος υπολογισμού των αριθμών Fibonacci: Αρχικά θέτουμε F(0) = 0, F(1) = 1 και έχουμε F(n) = F(n-1) + F(n-2). Για τη σημασία των αριθμών αυτών δείτε πχ στη Wikipedia.

35. Να γραφεί μια αναδρομική μέθοδος υπολογισμού του N! (για N μικρό φυσικά, πχ μέχρι 5).

36. Να μελετήσετε την ακολουθία μικρο-εντολών που δίνεται παρακάτω και να γράψετε τον αντίστοιχο RTL κώδικα. Για ποια εντολή IJVM/JAS πρόκειται;

```
0x04  0x05 000 00110110 000000001 010 0100
0x05  0x06 000 00010100 100000000 000 0111
0x06  0x02 000 00111111 001000010 100 0000
```

37. Γράψτε το μικρο-κώδικα (δυαδική μορφή) που αντιστοιχεί στην εντολή ISTORE . Ποιά σημεία του μικρο-κώδικα αλλάζουν για την εντολή ILOAD;

38. Γράψτε μικρο-κώδικα MAL για να υλοποιήσετε τις αριθμητικές - λογικές εντολές

    INEG : pop a, -a, push -a
    IINV : pop a, NOTa, push a

    Δοκιμάστε τη λειτουργία των νέων εντολών σας με τη βοήθεια μικρών προγραμμάτων ελέγχου σε γλώσσα JAS.

39. Γράψτε μικρο-κώδικα MAL για να υλοποιήσετε τις δύο από τις εντολές IF

    IFGT offset if (>0) goto offset
    IFLE offset if (<=0) goto offset
    IFGE offset if (>=0) goto offset

    IF_ICMPGT offset if (x > y) goto offset
    IF_ICMPGE offset if (x >= y) goto offset
    IF_ICMPLT offset if (x < y) goto offset
    IF_ICMPGE offset if (x <= y) goto offset

    Δοκιμάστε τη λειτουργία των νέων εντολών σας με τη βοήθεια μικρών προγραμμάτων ελέγχου σε γλώσσα JAS.

40. Στο πρόγραμμα που ακολουθεί συγκρίνετε τον αριθμό κύκλων μηχανής που απαιτούνται για την εκτέλεσή του με βάση τις μικρο-αρχιτεκτονικές Mic-1, Mic-2.

```
        BIPUSH 3
        ISTORE i
        BIPUSH 1
        ISTORE x
L1:     ILOAD i
        IFLT L2
        ILOAD x
        DUP
        IADD
        ISTORE x
        IINC i -1
        GOTO L1
L2:     ...
```

41. Το ίδιο όπως στην άσκηση 1.

```
        BIPUSH 0
        ISTORE x
        BIPUSH 0
```

```
        ISTORE i
L1:     BIPUSH 19
        ILOAD i
        ISUB
        IFLT L2
        ILOAD x
        ILOAD i
        IADD
        ISTORE x
        IINC i 1
        GOTO L1
L2:     ...
```

42. Με βάση τη μελέτη για τη ποσοστιαία συμμετοχή των εντολών επιπέδου ISA που συζητήσαμε στην εισαγωγή, υπολογίστε τη μέση ποσοστιαία βελτίωση της απόδοσης ενός υπολογιστή που διατηρεί το ίδιο επίπεδο ISA (δηλ τη IJVM) αλλά αλλάζει το επίπεδο μικρο-αρχιτεκτονικής του, από Mic-1 σε Mic-2.

43. Με βάση το σχέδιο μικρο-εντολών Mic-2 γράψτε ενδεικτικά σε δυαδική μορφή ορισμένες ακολουθίες μικρο-εντολών, για παράδειγμα της ILOAD και ISTORE.

44. Ανασχεδιάστε την μικρο-αρχιτεκτονική Mic-1 ώστε να εκτελεί τις μικρο-εντολές Mic-2. Απαιτούνται αλλαγές στο χρονισμό του συστήματος;

45. Με βάση τη μικρο-αρχιτεκτονική διοχέτευσης Mic-3 μετατρέψτε την απλή RTL/MAL που ακολουθεί σε RTL με διοχέτευση. Κάθε γραμμή αντιστοιχεί σε μια μικρο-εντολή RTL/MAL. Εξηγείστε τη διαδικασία μετατροπής. Για ποια εντολή γλώσσας μηχανής IJVM/JAS πρόκειται; Ποια είναι η βελτίωση της απόδοσης αν ο χρονισμός στη CPU με διοχέτευση είναι 1/3 του χρόνου της CPU χωρίς διοχέτευση; Τι φαινόμενο παρατηρείτε στη συγκεκριμένη διοχέτευση; Πού οφείλεται;

```
MAR=SP=SP-1;rd
H=TOS
MDR=TOS=MDR+H;wr;
```

46. Γράψτε την ακολουθία μικρο-εντολών για την εκτέλεση των παρακάτω αποσπασμάτων JAS στην μικρο-αρχιτεκτονική διοχέτευσης Mic-3. Υπολογίστε τη επιτάχυνση σε σχέση με την εκτέλεση σε Mic-2 καθώς και την ωφελιμότητα.

```
ILOAD x
DUP
IADD
ISTORE x
IINC i -1
....
ILOAD x
ILOAD i
ISUB
ISTORE x
IINC i 1
```

47. Γράψτε πρώτα το κώδικα JAS και κατόπιν την ακολουθία μικρο-εντολών για την εκτέλεση του παρακάτω αποσπάσματος κώδικα Java/C σε μικρο-αρχιτεκτονική διοχέτευσης Mic-3, στη περίπτωση TRUE και FALSE. Υπολογίστε επιτάχυνση και ωφελιμότητα.

IF (i == 3) k = 0 else k = 1;

48. Έστω ότι έχουμε στη διάθεσή μας μνήμη RAM μεγέθους 2Gbytes και η μνήμη μας είναι byte-addressable. Έστω οτι έχουμε στη διάθεσή μας 1Mbyte cache και το μέγεθος του block (ή γραμμής) είναι 64 bytes.

    Πόσα bits απαιτούνται για τη φυσική/λογική διεύθυνση;

    Πόσες είναι οι διαθέσιμες γραμμές (blocks) της κρυφής μνήμης; Στη μέτρηση λαμβάνουμε υπ' όψη μόνο το χώρο που καταλαμβάνουν τα δεδομένα, όχι οι υπόλοιπες πληροφορίες της κρυφής μνήμης.

    Σχεδιάστε τη δομή της λογικής διεύθυνσης σε περίπτωση που η οργάνωση της κρυφής μνήμης ειναι: (α) 'Αμεσης Χαρτογράφησης (β) Συνολοσυνειρμική 2 Δρόμων και (γ) Συνολοσυνειρμική 4 Δρόμων

49. Έστω τα πρώτα 16 blocks της κύριας μνήμης (ξεκινάμε από τη μηδενική διεύθυνση). Γράψτε τη διευθύνσης της αρχής κάθε block. Για κάθε μια από τις παραπάνω οργανώσεις κρυφής μνήμης σημειώστε σε ποιές θέσεις της κρυφής μνήμης μπορεί να αποθηκευτεί το κάθε ένα από τα 16 blocks.

50. Έστω μια κρυφή μνήμη 8 γραμμών, όπου κάθε γραμμή μπορεί να αποθηκεύσει 64 λέξεις των 2 bytes. Η κύρια μνήμη έχει συνολικό μέγεθος 4Kbytes.

    Σχεδιάστε την εικονική διεύθυνση για τις περιπτώσεις (α) μνήμης άμεσης χαρτογράφησης (β) συνολοσυνειρμικής μνήμης 2 δρόμων (γ) πλήρως συνειρμικής μνήμης.

    Δείξτε την αντιστοίχιση γραμμών κύριας μνήμης με γραμμές κρυφής μνήμης στις τρείς παραπάνω περιπτώσεις. Πόσες είναι οι γραμμές της κύριας μνήμης;

    Εκτελέστε την λειτουργία ανάγνωσης μιας λέξης για την φυσική διεύθυνση 520 (δεκαδικό) στις τρείς παραπάνω περιπτώσεις.

51. Προτείνετε ένα δικό σας σχήμα κωδικοποίησης των εντολών για το επίπεδο ISA της IJVM για το Mic-4. Αυτό που αλλάζει είναι η χρήση 8 καταχωρητών (R0-R7) στη θέση του stack.

52. Η άσκηση που ακολουθεί είναι σχετικά δύσκολη. Έστω οτι θέλουμε να επεκτείνουμε τη λειτουργία του Mic-1 ώστε να καλύπτει λειτουργίες δεικτών και πινάκων. Για να το κάνουμε αυτό πρέπει να επεκτείνουμε τη JAS και την MAL ανάλογα. Ενδεικτικά πρέπει να μπορούμε να γράψουμε εντολές όπως

```
int x, t, a[10];
int *y;
...

y = &t;
x = *y;
*y = x;

a_method(int *y);

*y++;

a[i] = 5
*(a+i) = 5;
```

    Σχεδιάστε εντολές JAS και υλοποιείστε τις σε MAL για τις παραπάνω λειτουργίες. Ενδεικτικά προτείνονται οι εντολές:

ALOAD t φόρτωση στη στοίβα της διεύθυνσης της μεταβλητής t

ILOAD y φόρτωση στη στοίβα τιμής της μεταβλητής με διεύθυνση που βρίσκεται στη μεταβλητή y
ISTORE y αποθήκευση της τιμής της στοίβας στη μεταβλητή με διεύθυνση που βρίσκεται στη μεταβλητή y

IALOAD a, i φόρτωση στη στοίβα της τιμής που βρίσκεται στη θέση μνήμης *(a+i)
IASTORE a, i αποθήκευση της τιμής της στοίβας στη θέση μνήμης *(a+i)