ΠΡΟΗΓΜΕΝΟΙ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ
PROJECT 2: MEMORY MANAGEMENT

ΘΕΩΡΙΑ

Στο project αυτό έχουμε υλοποιήσει τις βασικές συναρτήσεις της stdlib της C malloc και free. Η συνάρτηση malloc είναι η void *malloc(int size) η οποία δεσμεύει και επιστρέφει περιοχή μνήμης μεγέθους size. Η συνάρτηση void free(void *mem_ptr) η οποία αποδεσμεύει την περιοχή μνήμης που δείχνει ο pointer. Ο pointer παίρνει τιμές που επιστρέφει η συνάρτηση malloc. Οι συναρτήσεις malloc και free έχουν ευρεία χρήση καθώς επιτρέπουν τη δυναμική δέσμευση και αποδέσμευση μνήμης σε περιπτώσεις που δεν ξέρουμε κατά το compile time τον ακριβή χώρο μνήμης που χρειάζεται να δεσμεύσουμε. Για παράδειγμα ένα πρόγραμμα που κρατάει βιβλίο διευθύνσεων στο οποίο θα πρέπει να μπορούμε να προσθέσουμε απεριόριστες εγγραφές για κάθε εγγραφή που προσθέτουμε δυναμικά δεσμεύουμε τον επιπλέον χώρο που χρειάζεται.

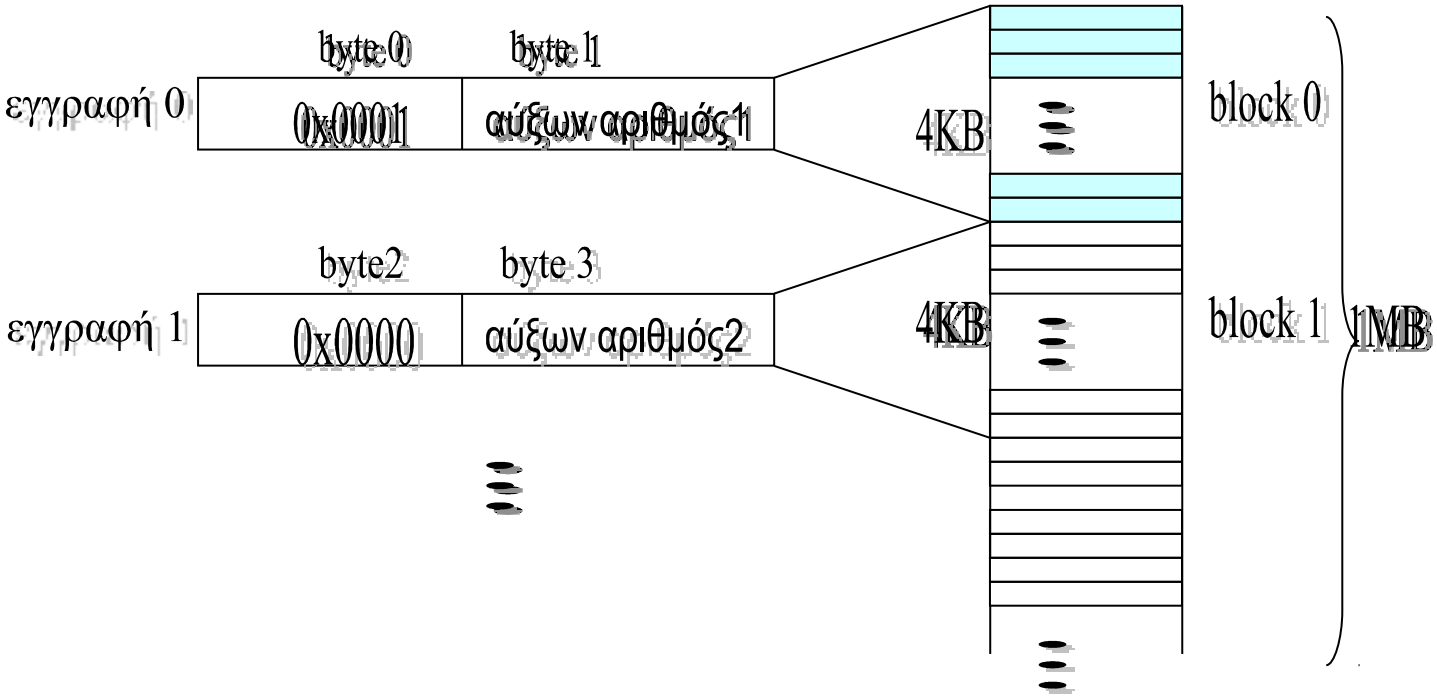
ΥΛΟΠΟΙΗΣΗ

Οι συναρτήσεις malloc, free έχουν χρησιμοποιηθεί από την άσκηση 5(task switching) όπου δεσμεύουμε δυναμικά το χώρο για τα δεδομένα που απαιτεί η άσκηση σε αντίθεση με την πρωτότυπη άσκηση που δέσμευε το χώρο στατικά.

Αρχικά, μετά το τέλος του αρχείου της άσκησης έχουμε δεσμεύσει 1MB για δυναμική μνήμη και επιπλέον 512 bytes που χρησιμοποιούνται για ένα πίνακα ανάθεσης που κρατά πληροφορίες για τη μνήμη που δεσμεύσαμε. Ο πίνακας αυτός έχει την παρακάτω μορφή:

	0: αδέσμευτο, 1: δεσμευμένο	Άυξων αριθμός
1η Εγγραφή	byte0	byte1
2η Εγγραφή		
.	.	.
.	.	.
.	.	.
256η Εγγραφή		

Το 1MB μνήμης χωρίζεται σε 256 κομμάτια των 4KB. Κάθε εγγραφή του πίνακα ανάθεσης κρατά πληροφορίες για κάθε κομμάτι των 4KB δυναμικής μνήμης και η αντιστοιχία φαίνεται παρακάτω:



Κάθε εγγραφή του πίνακα είναι 2 bytes, το λιγότερο σημαντικό byte δείχνει αν το κομμάτι μνήμης είναι δεσμευμένο ή όχι και το περισσότερο σημαντικό έναν μοναδικό αύξοντα αριθμό για το κάθε κομμάτι που δεσμεύουμε.

Υλοποίηση malloc

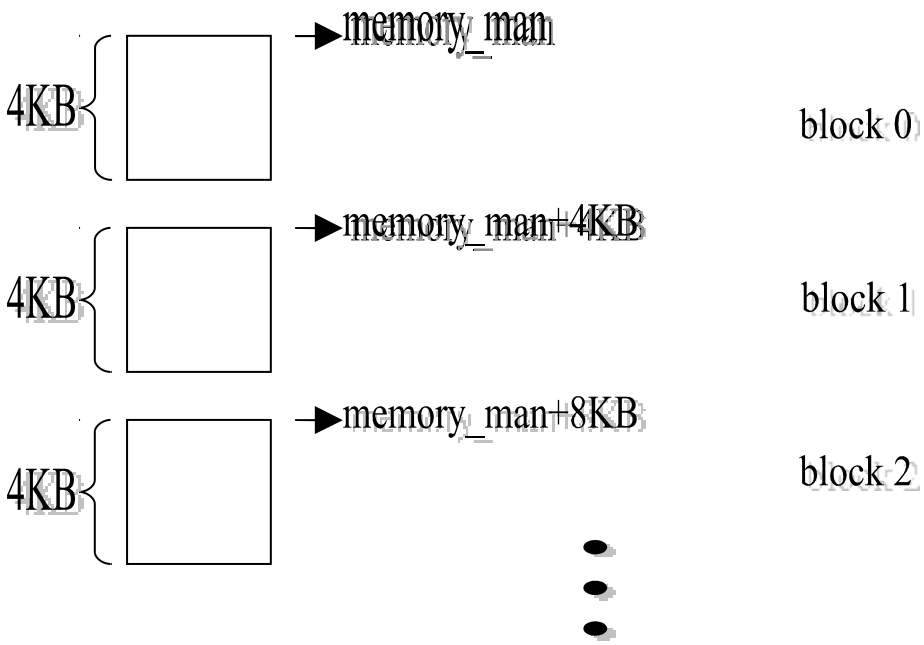
Για την υλοποίηση της malloc εξετάζουμε αν το όρισμα που παίρνει είναι πολλαπλάσιο των 4KB. Στην περίπτωση αυτή έχουμε ακριβώς τα κομμάτια των 4KB που πρέπει να δεσμεύσουμε. Αν το όρισμα δεν είναι πολλαπλάσιο των 4KB χρησιμοποιούμε το αμέσως επόμενο πολλαπλάσιο το οποίο μας δίνει τον αριθμό των κομματιών που τελικά θα δεσμεύσουμε. (Τον αριθμό αυτό τον βρίσκουμε προσθέτοντας σε αυτόν τον αριθμό 4095=(4K - 1) και μηδενίζοντας τα 10 λιγότερο σημαντικά bits του).

Στη συνέχεια καλούμε τη συνάρτηση search η οποία ψάχνει στον πίνακα ανάθεσης για συνεχόμενα αδέσμευτα κομμάτια των 4KB. Αυτή η συνάρτηση παίρνει σαν ορίσματα την αρχή του πίνακα (memory_matrix) και το μέγεθος του πίνακα (matrix_size). Το κυρίως σώμα της search ελέγχει το λιγότερο σημαντικό byte της κάθε εγγραφής του πίνακα ανάθεσης και αν είναι αδέσμευτη συνεχίζει τον έλεγχο στην επόμενη

μέχρι να βρει όλες τις διαδοχικές αδέσμευτες θέσεις. Όταν τελειώσει αυτή η διαδικασία η συνάρτηση μας επιστρέφει τον αριθμό των συνεχόμενων αδέσμευτων θέσεων(ebx), την πρώτη εγγραφή του πίνακα που δείχνει σε αδέσμευτο κομμάτι (edx) και τον αριθμό των θέσεων του πίνακα που μένει να ελεγχθεί (ecx).
Μετά την κλήση της search ελέγχουμε αν ο αριθμός των αδέσμευτων κομματιών που βρήκαμε είναι μεγαλύτερος ή ίσος από αυτόν που χρειαζόμαστε. Αν ισχύει αυτό δεσμεύουμε το μέγεθος που θέλουμε, διαφορετικά ξανακαλούμε τη search μέχρι να τελειώσει ο έλεγχος ολόκληρου του πίνακα. Αν δεν καταφέρουμε να βρούμε το μέγεθος που ζητάμε, η συνάρτηση επιστρέφει το 0. Αν βρούμε το μέγεθος που ζητάμε πηγαίνουμε στον πίνακα ανάθεσης και ξεκινώντας από την πρώτη εγγραφή που δείχνει στο αδέσμευτο αυτό κομμάτι αποθηκεύουμε στο λιγότερο σημαντικό byte τον αριθμό 1 (χαρακτηρίζει τα δεσμευμένα κομμάτια) και στο περισσότερο σημαντικό έναν μοναδικό αύξοντα αριθμό. Αυτή η διαδικασία γίνεται για όλες τις εγγραφές που απαιτεί το μέγεθος.
Τέλος αντιστοιχίζουμε την εγγραφή του πίνακα που αναφέρεται στο πρώτο κομμάτι που δεσμεύσαμε στην πρώτη θέση φυσικής μνήμης που θα χρησιμοποιήσουμε. Για να το πετύχουμε αυτό αρχικά διαιρούμε τη διεύθυνση της πρώτης θέσης του πίνακα που δείχνει σε αδέσμευτη θέση μνήμης με 2 ώστε να βρούμε το μπλοκ της μνήμης από το οποίο θα αρχίσουμε τη δέσμευση.Στη συνέχεια πολλαπλασιάζουμε τον αριθμό αυτό με το 2^{12} (=4Kbytes) και τον προσθέτουμε στην πρώτη διεύθυνση της μνήμης που έχει δεσμευτεί για δυναμική διαχείριση ώστε να πάρουμε τη διεύθυνση της φυσικής μνήμης απ' όπου θα αρχίσουμε να δεσμεύουμε. Τώρα η malloc επιστρέφει τη διεύθυνση της θέσης αυτής.

Υλοποίηση free

Η συνάρτηση free χρησιμοποιεί ως ορίσματα τις τιμές που επιστρέφει η malloc έτσι ώστε να αποδεσμεύσει τα αντίστοιχα κομμάτια μνήμης που έχουν τον ίδιο αύξοντα αριθμό (δεσμεύτηκαν στην ίδια κλήση της malloc). Η μορφή της μνήμης είναι η εξής:



Έχοντας λοιπόν τώρα περάσει ως όρισμα στη συνάρτηση free την πρώτη διεύθυνση της φυσικής μνήμης που θα αποδεσμεύσουμε, αφαιρούμε από αυτή την αρχή της μνήμης(memory_man) και στη συνέχεια τη διαιρούμε με 2^{12} (=4Kbytes) ώστε να πάρουμε τον αριθμό του block από το οποίο θα αρχίσει η αποδέσμευση. Τέλος πολλαπλασιάζουμε τον παραπάνω αριθμό με 2 για να βρούμε σε ποια εγγραφή του πίνακα αναφερόμαστε.
Αν για παράδειγμα θεωρήσουμε ότι το όρισμα που παίρνει η free είναι το mem_man+8KB ακολουθώντας την παραπάνω διαδικασία το μπλοκ της μνήμης από το οποίο θα αρχίσει η αποδέσμευση είναι το μπλοκ 2 το οποίο με την παραπάνω διαδικασία αντιστοιχίζεται στην εγγραφή 2 του πίνακα (δηλαδή στην $3^{η}$ κατά σειρά εγγραφή του πίνακα).
Αφού κρατήσουμε τον αύξοντα αριθμό της εγγραφής αυτής ξεκινάμε από εκεί την αποδέσμευση μηδενίζοντας το λιγότερο σημαντικό byte της κάθε επόμενης εγγραφής. Η αποδέσμευση σταματάει όταν συναντήσουμε κάποια εγγραφή με διαφορετικό αύξοντα αριθμό ή όταν φτάσουμε στο τέλος του πίνακα.

Κώδικας παρουσίασης

Ο κώδικας παρουσίασης έχει αναπτυχθεί πάνω στον κώδικα της άσκησης 5 (task switching).
Τις συναρτήσεις malloc, free τις έχουμε χρησιμοποιήσει στα πλαίσια της άσκησης 5 για να δεσμεύσουμε δυναμικά κομμάτια μνήμης όπου αυτό απαιτείται αλλά έχουμε αναπτύξει και ένα ξεχωριστό κώδικα παρουσίασης το οποίο επιβεβαιώνουμε την ορθή λειτουργία των συναρτήσεων.

Στον κώδικα αυτό αρχικά καλούμε 3 συνεχόμενες φορές τη malloc με όρισμα:

1. 4097(ζητώντας της δηλαδή να δεσμεύσει 2 blocks των 4KB και η τιμή που επιστρέφει αποθηκεύεται στον ebx)
2. 12200(ζητώντας της να δεσμεύσει 3 blocks, αποθηκεύοντας την τιμή που επιστρέφει στον ecx)
3. 5000(ζητώντας της να δεσμεύσει ξανά 2 blocks, αποθηκεύοντας την τιμή που επιστρέφει στον edx)

Παρακάτω στον κώδικα παρουσίασης καλούμε τη συνάρτηση free περνώντας της ως όρισμα τον ecx. Τέλος ξανακαλούμε τη συνάρτηση malloc με ορίσματα:

1. 16380(ζητώντας της δηλαδή να δεσμεύσει 4 blocks και η τιμή που επιστρέφει αποθηκεύεται στον esi)
2. 4096(ζητώντας τώρα να δεσμεύσει 1 block και η τιμή επιστρέφεται στον eax)

Μετά από την εκτέλεση του παραπάνω κώδικα τα αποτελέσματα των καταχωρητών που προκύπτουν είναι τα εξής:

Eax=0x00002af8
Ebx=0x00000af8
Ecx=0x00002af8
Edx=0x00005af8
Esi=0x00007af8

Παρατηρούμε ότι το κομμάτι που αποδεσμεύεται από τη free δεν επαρκεί για τη δέσμευση που απαιτεί η αμέσως επόμενη malloc. Έτσι αυτή η malloc επιστρέφει τη διεύθυνση 0x00007af8 και όχι τη 0x00002af8. Ενώ η τελευταία malloc επιστρέφει τη διεύθυνση 0x00002af8 γιατί τώρα το κομμάτι αυτό που είχαμε αποδεσμεύσει επαρκεί.

Στον κυρίως κώδικα της άσκησης 5 χρησιμοποιούμε τη malloc για δεσμεύσουμε δυναμικά χώρο για τις μεταβλητές timer, active1, active2. Τις διευθύνσεις που επιστρέφει η malloc τις φυλάμε στους καταχωρητές edi, esi τους οποίους χρησιμοποιούμε σαν δείκτες για να προσπελάσουμε τις μεταβλητές αυτές (τους καταχωρητές των task1 και task2 τους ενημερώνουμε γράφοντας στα αντίστοιχα πεδία των tss) .