עבודה 2 באת"מ

תיעוד חיצוני

**מבוא**

דמקה הוא משחק לוח שבו פרוסים על לוח משבצות אבנים בשתי צבעים כאשר מטרת המשחק להוריד מהלוח את כל אבני היריב על ידי "אכילתן", בכל טור על השחקן לקדם באלכסון את אחד מאבניו או לחלופין במידה ויכול לקפוץ באלכסון מעל אבן היריב ו"לאכול" אותה.

**מטרת התוכנית**

מטרת התוכנית היא לחשב בכל מצב נתון של הלוח את הצעד הטוב ביותר שיכול שחקן מסיום לעשות לפי שיטת המינימקס. שיטת המינימקס באה מתורת המשחקים נקראת כך כיוון שכל שחקן שואף למקסם את התשלום המינימלי שהוא יכול לקבל מהמשחק, או למזער את ההפסד המקסימלי.

**הסבר של תוויות ושימוש של הרגיסטרים**

שימושים בריגיסטרים :

רגיסטר 0 – מונה של מספר הצעדים החוקיים

רגיסטר 2 – מיקום נוכחי בלוח לבדיקת אפשרויות תנועה

רגיסטר 3 – מיקום בעמודה כאשר 0 הוא עמודה שמאלית(המיקום בשורה)

רגיסטר 5 – מספר שורה כאשר 0 שורה עליונה, ירידה של שורה נגרמת על ידי הגדלה של 20

רגיסטר 4 – מיקום זמני כדי לבדוק אפשרות אכילה מבלי להתעסק עם רגיסטר 2

\*\*\*קיימים עוד שימושים שלהם בשגרות השונות

הסבר:

**getMaxWP**

שגרת מעטפת המאתחלת את המחסנית, משתמשת ב findMv לקבלת המהלך האופטימאלי ונתוניו, מעבירה בעזרת המחסנית את הערכים שקיבלנו ל SrcPos DstPos WinParam בהתאמה.

**edge cases**

**hasMoves**

**CalcWP**

**checkWin**

**minOrMax**

minOrMax

משווה בין השחקן הנוכחי לשחקן המקורי שקיבלנו (לפי כתובתו בזיכרון). אם זהים נעביר 1 לרגיסטר 4(כלומר השחקן הנוכחי זה השחקן שקיבלנו ולכן נחפש לו תוצאה מקסימלית). אם אינם זהים נעביר 1- לרגיסטר 4. בכל מקרה נעבור ל eminOMax.

eminOMax

חוזר לשגרה

**movPiece\***

**reverse movPiece\***

**cmpWP**

**recursion**

**finding number of moves or pieces**

\* funAct

תווית הבודקת מה צריך לבצע לפי funUse. אם funUse שווה 1 מונה את מספר הצעדים. אם לא (כלומר funUse שווה ל2) עובר לרקורסיה.

numOmoves

מעביר את האיבר השלישי במחסנית לfunUse . שומר את ערכי רגיסטרים 0 עד 5 במחסנית ומאתחל אותם.

startMv

פעולה שמכניסה לרגיסטר השני את מיקום הבדיקה הבאה בלוח עמודה (רגיסטר 3) ושורה (רגיסטר 5).אם הגענו לסוף הלוח עובר ל.JendcodeM אם לא בודק לאיזה שחקן אנחנו רוצים לספור את מספר הצעדים האפשריים שחור (2) או לבן (1) ועובר לתווית המתאימה בהתאם (whtsMv או blksMv ).

JendcodeM

קופץ לendcodeM (נמצא למטה)

blksMv

חל שימוש בתווית זו כאשר מחפשים את מספר האפשרויות לאבן שחורה (נקבע ב (startMv. בודק אם במיקום הוכנס בריגיסטר 2 יש שחקן בצבע שחור – אם כן ממשיך כדי לבדוק את הצעדים האפשריים אם לא עובר הלאה. בודק אם funUse שווה ל2 אם כן עובר ל.JincPiectאם לא בודק האם קיימת אפשרות לביצוע צעד חוקי לכיוון שמאל (אם צעד רגיל או אכילה לשמאל) מוסיף אחד לרגיסטר 0 ועובר לתווית blksMv2

JincPiec

קופץ לincPiec (נמצא למטה)

blksMv2

אם קיימת אפשרות לביצוע צעד חוקי לכיוון ימין (אם צעד רגיל או אכילה לימין) מוסיף 1 לרגיסטר 0

advncMv

עובר למיקום הבא לבדיקת מספר הצעדים החוקיים, אם המיקום הנוכחי הוא בקצה השורה משתמש ב incrow למיקום הבא לבדיקה.

incrow

עובר לשורה הבאה לבדיקה.

whtsMv

המקביל של blksMv ללבנים בהתאמה.

whts2Mv

המקביל של blksMv2 רק ללבנים בהתאמה.

eatblksMv

אם ניתן לבצע אכילה לשמאל מוסיף 1 לרגיסטר 0 וקופץ לblksMv2 בכל מקרה.

eatblksMv2

אם ניתן לבצע אכילה לימין מוסיף 1 לרגיסטר 0 וקופץ ל advncMv בכל מקרה.

eatwhtsMv

המקביל של eatblksMv ללבנים בהתאמה.

eatwhtsMv2

המקביל של 2eatblksMv ללבנים בהתאמה.

endcodeM

חל שימוש בתווית זו לאחר שעברנו על כל המשבצות בלוח המשחק. משווה את funUse ל0 . אם שווה ל0 עובר ל ecodeM . אם לא מעביר 0 ל funUse ,מעביר מרגיסטר 0 לאיבר השני מלמעלה במחסנית את מספר הצעדים החוקיים האפשריים לשחקן שהתבקש לבדוק (לבן או שחור 1 או 2) ואז עובר ל ecodeM .

ecodeM

חזרה מהשגרה

incPiec

מגדיל את רגיסטר 0 ב1 וקופץ ל advncMv

stepsAdd

מגדיל את רגיסטר 0 (מספר הצעדים) ב1 וחוזר מהשגרה

**recursion**

**change Player**

cngPlayer

מחליף שחקן- מכניס את מספר השחקן החדש בראש המחסנית וב currPl. יוצא מהשגרה

**reverse Colors**

cngClr

מחליף צבע (שחקן)- מחליף את צבע השחקן שנמצא באיבר השני מלמעלה במחסנית.