

בשילוב הקודים האלו בדרך כלל נעשה "מזיקה" באמצעות forwarding unit  
פתרון זה סוגר את הפקודות כמו Add, or וכו'. אם לא, הפקודה  
שנ"ח המזיקה נשלח ישר אל הבקרה וללא הזיכרון הבין א"ן  
לפי את המדק (ולכן לא נ"ח) והפקודה נשלח אל המזיכרון הבין, אבל  
(לא) להשתמש בפקודת Nop.

הקומפליקציה של מומיאר אשר מבוצעת אינטימציה, הקומפליקציה מקשה  
עשירה את סדר הפקודות מאפי אפילו בקוד ובכך אומץ הפקודות Nop.

כדי שבפקודת נ"ח ובקוד Nop ברמה המזיקה, אין מוססם וחיה  
זוהי מזיקה (Hazard). בעת פקודת נ"ח, היתירה היבט של הפקודה  
הבאה נ"ח (או bubble) ל"ח הפסקת השלשון לזיכרון הפקודות והצורה  
ק"י הפקודה נ"ח לזיכרון כתירה לרעסטרס.

כך כה סיפוף - Data hazard. ע"י, וזה לפתור את בעיית ה- branch.  
הבעיה ה"ח שלב שהמחשב "מחנן" ששלוני אבד branch, (בנסיה 2 פקודות  
ע"י ח"ב ה- pipeline.

כידק, נעשה לו branch hazard 2 תחילה, (ולכן במחנן להשתמש בהוספה 3 פקודות  
Nop.

אינטימציה ר"שית - נעשה את א פקודת ה- branch ע"ימזיק השני, כותר, הפולח  
תזדויל כעת Nop את כלב כ כך בשלב השני יעדינו הכתובה של  
הפקודה הבאה.

אופטימציה של"ח - branch-delay slot - כאשר נעשה הפקודה לאחר ה- branch תמיד  
מבצעת, כן, אם א"ח מה אבד כל יקרה, נעשה Nop. את זה יש, נ"ח עסקים  
שורה אחת ה- branch וכן חסוק מ"חן. הפקודה שנעשה, ר"ח כ- 50%  
המזיקים יש פקודה שנתן עסקים ובכך חסוק מ"חן.

אחד הפרמטרים המשמעותיים ברובם מחשב הקוד הוא כמות החשבוש שהיא זיכרון או מ"חן חסוק  
שיטת ה- pipeline, (א) א over-head שלכך בה, זיכרון יותר חשבוש ומזיקה  
יותר ח"ח.

כך אבדו bubble בשלב השלשון, אפסם קו flush אשר אפסם א ה- Nop  
code א"ח וזי כן, כמקרה של branch חסוק את הפקודה שנעשה  
פקודת Nop.

חלום ה-CPU - פקודת נ"ח חלוקה ג"ס, (נ"ח ע"ש) א"ח ר"חן הש"ח א"ח ר"ח  
המפלה, ויזרה ח"ח, בעת החלפת ח"ח, ה- CPU אשתמש בקו"י בקוד  
כדי אפוק א bubble את הפקודות הבאות כמו כן, 2 ריגיסטרים מ"חן  
שומרים את ע"ח השלשון והכתובה של הפקודה הבאה בקו"י, אשתמש  
עסקה שנקבטה לראש, מבקלים ע"חן המזיקות ואנפסם את השלשון  
השכרה ה"ח נכבד ע"י ח"חן וזקו"ח ש"חן בר"ח. נ"ח. בשלב ה-1.

זיכרון אחר אמנוש ק"י הפקודה, במקום אשתמש בשלשון AND ר"ח, (אשתמש בח"חן  
זיכרון, אשר הכתובה ש"חן א"ח יהיה פקודה ה- code ש"חן וזמ"חן ח"ח  
ש"חן ק"י הפקודה (א"ח) יה"ח ש"חן בזיכרון). לזיכרון זה הוא אפוק ROM  
ש"חן יותר קטן וז"ח מזיכרון המשפטר ג"ס כתובה.

בשנים האחרונות, ח"חן (א"ח) הח"חן בין המחירות של ה CPU אשתמש של הזיכרון  
ח"חן המשקנים של הפקודה (ש"חן בזיכרון, א"ח) א"ח (ש"חן את הזיכרון, הח"חן  
ה"חן כ"ח א"ח א"ח.

אחת הע"חן אשתמש א"ח כן, הוא באמצעות מחנן ("cache"). לז"ח זיכרון ר"ח

יותר מהיר שכן המערכת זוכה במידע שהיא מחפשת ואין צורך לחפש אותו באיזורי זיכרון אחרים.  
כמו כן, המערכת זוכה במידע שאיננו זקוקים ממשיך ב-cache ואין צורך לשלוח אותו לזיכרון.

הנהגת ה-Pipeline

יש 2 סוגי הנהגת ה-Pipeline: סטטיקה ו- "דאנטינג" (מיון). הם מחפשים בראשית דבר  
לחבר את ה- CPU והמערכת למה שזקוקים להם במידע. הנהגת ה-Pipeline היא רציפה או אינטרקטיבית.  
ה- CPU והמערכת לא רצים במקביל אלא הם מחפשים זה את זה.

RAM

יש 2 סוגי RAM: RAM סטטיקה ו-RAM דינמית. ה-RAM סטטיקה היא RAM חדישה יותר  
וה-RAM דינמית היא RAM שהיא זולה יותר והיא RAM חדישה פחות.

D-RAM סוגי RAM דינמית. במידע דינמית יש צורך להעביר את המידע לזיכרון זמני.  
מבחינתם, המידע נשמר במידע דינמית.

SRAM - קניי ה-D-FF הוא המהיר שיש והוא המהיר שיש. למה? כי הוא פשוט יותר.  
יש זיכרון זמני המכיל את המידע. ה-RAM דינמית היא זולה יותר והיא זולה יותר.  
ה-RAM דינמית היא זולה יותר והיא זולה יותר.

הזיכרון ה- DRAM, יש צורך להעביר את המידע לזיכרון זמני.  
הוא זולה יותר והוא זולה יותר.

Hit - מצב בו המידע שנדרש נמצא בזיכרון המהיר.

Miss - מצב בו המידע שנדרש אינו נמצא בזיכרון המהיר.

Block - יש צורך להעביר את המידע לזיכרון זמני.

יש מספר סוגי Miss: compulsory, conflict, capacity. הם מיונים שונים של המידע.  
compulsory: מיון המידע שצריך להיות בזיכרון המהיר.  
conflict: מיון המידע שצריך להיות בזיכרון המהיר.  
capacity: מיון המידע שצריך להיות בזיכרון המהיר.

Direct Mapped Cache - שיטה לתחזוקת הזיכרון המהיר.  
קובעת באיזה זיכרון המידע צריך להיות.  
יש זיכרון זמני המכיל את המידע. ה-Direct Mapped Cache היא שיטה לתחזוקת הזיכרון המהיר.

write buffer - שיטה לתחזוקת הזיכרון המהיר.  
היא זולה יותר והיא זולה יותר.  
יש זיכרון זמני המכיל את המידע. ה-write buffer היא שיטה לתחזוקת הזיכרון המהיר.