

Computersystemen en -architectuur

MIPS Project: Deel 2

1 Ba INF
2025–2026

Tim Apers

Assistent

tim.apers@uantwerpen.be

Tibo Verreycken

Student-assistent

tibo.verreycken@student.uantwerpen.be

Maandag 1 december 2025

1 Introductie

In dit document vind je de uitleg voor het tweede deel van het MIPS project. De algemene uitleg van het project kan je lezen in de PDF van deel 1.

2 Deel 2: werkend videospel

2.1 Indienen en beoordeling

Je dient de opdracht in tegen **maandag 15 december 2025, 22:00**. Je dient je bestand **game.asm** in via Inginious: “CSA” → “MIPS” → “Mips Project Deel 2”. Tijdens de beoordeling zal de code uitgevoerd worden met MARS om te kijken of je programma correct werkt. Zorg er dus voor dat je programma’s **correct werken in MARS!** Voor de beoordeling wordt de volgende puntentabel gehanteerd:

Onderdeel	Score
Code kwaliteit	4
Correct weergeven van het doolhof	6
Doolhof wordt dynamisch ingeladen en is niet gehardcode	6
De speler kan door wandelgangen bewegen	6
De speler kan niet door muren bewegen	6
Als de speler de uitgang bereikt eindigt het spel	6
Gebruik van stackframes	6
Totaal	40

Tabel 1: Puntentabel voor het tweede deel.

2.2 Opdracht

Voor het tweede onderdeel zal je een werkend videospel implementeren, gebruikmakend van de onderdelen en de kennis die je tijdens het vorige deel hebt opgedaan. Op MSDL vindt je een template file **game.asm** die je kan gebruiken om je videospel te implementeren. In deze file vindt je enkele nuttige elementen in het **.data** gedeelte die je kan gebruiken voor de implementatie. Voeg voldoende comments toe aan je code en organiseer je code in functies met behulp van stackframes.

2.2.1 Inlezen van het doolhof

Schrijf een functie dat een doolhof inleest uit een bestand:

- Elke letter in het bestand stelt één vakje in het doolhof voor. De volgende codering is van toepassing:

Character	Betekenis	Kleur
w	muur	blauw
p	doorgang	zwart
s	speler	geel
u	uitgang	groen
newline	eind van rij vakken	-

- De enige plaats in het geheugen waar het doolhof moet worden bijgehouden is het bitmap-geheugen. Elk vak van het doolhof wordt dus enkel als een gekleurde pixel bijgehouden.
- De speler kan niet door muren (in het blauw) en kan enkel de wandelen door de gangen (zwart).
- De positie van de speler wordt ook bijgehouden in twee registers: één voor het rijnummer en één voor het kolomnummer.
- Om deze rij- en kolomnummers om te zetten in de geheugenlocaties van de pixels kan je gebruikmaken van de functies van deel 1.
- De bovenste rij van het doolhof heeft rijnummer 0. De meest linkse kolom van het doolhof heeft kolomnummer 0. Voor een doolhof van 4 bij 5 vakken ziet dat er als volgt uit:

(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)
(4, 0)	(4, 1)	(4, 2)	(4, 3)

- Zoals reeds vermeld in de vorige opdracht, zal de pixel die overeenkomt met rij 0 en kolom 0 dus opgeslagen zijn op het geheugenadres in register **\$gp**.

2.2.2 Aanpassen van de positie van de speler

Schrijf een functie die de positie van de speler in het doolhof aanpast:

- De functie heeft 4 argumenten: de huidige positie van de speler (rij- en kolomnummer) en de nieuwe positie van de speler (rij- en kolomnummer). Gebruik hiervoor registers **\$a0-\$a3**.
- Om rij- en kolomnummers te vertalen naar de geheugenadressen van de pixels kan je gebruikmaken van de functies van deel 1.
- Er wordt eerst gecontroleerd of de nieuwe positie geldig is: controleer op de aanwezigheid van muren en dat de nieuwe positie binnen het speelveld valt.
- Pas de kleuren van de pixels aan.
- De nieuwe positie van de speler wordt gereturned door de functie met het nieuwe rijnummer in **\$v0** en het nieuwe kolomnummer in **\$v1**.

2.2.3 Main game loop

Schrijf een main functie met daarin een loop die het spel gaande houdt:

- De main loop is een eindeloze loop die de volgende stappen uitvoert:

1. Vraag de gebruiker om input. Gebruik daarvoor onderstaande codering:

Character	Actie
z	beweeg omhoog
s	beweeg omlaag
q	beweeg naar links
d	beweeg naar rechts
x	beëindig het spel

2. Controleer of er input is:

- (a) Indien er input is: ga naar stap 3
- (b) Indien er geen input is: ga naar stap 4

3. Voer de gepaste actie uit (aanpassen van de positie van de speler of afsluiten van het spel).

4. Wacht 60ms met behulp van de “sleep” `syscall`. Dit laat toe om het spel op een gepaste snelheid te spelen.

- Wanneer de speler de uitgang van het doolhof bereikt, wordt er een bericht geprint (met een `syscall`) en wordt het spel beëindigd.